# Advanced Computer Networking (ACN)

IN2097 – WiSe 2025–2026

**Prof. Dr.-Ing. Georg Carle, Sebastian Gallenmüller**

Christian Dietze, Marcel Kempf, Lorenz Lehle

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

# Motivation

## Reproducible experiments

- Everyone agrees that reproducible research is important
- Scientific community tries to incentivize reproducible research:

# Motivation

## Reproducible experiments

- Everyone agrees that reproducible research is important
- Scientific community tries to incentivize reproducible research:


# Motivation

## Problems with reproducibility

- Two workshops at SIGCOMM conference dedicated to reproducible research:
  - SIGCOMM'03: MoMeTools workshop
  - SIGCOMM'17: Reproducibility workshop
  - Problems remained the same over 14 years

## Best solution so far . . .

- Artifact Evaluation Committees & Reproducibility Badges
- Problems:
  - High effort
  - Potentially low robustness (CCR Apr. '20[1])



ACM's badges awarded by the Artifact Evaluation Committee

[1] [1] N. Zilberman, "An Artifact Evaluation of NDP", Comput. Commun. Rev., vol. 50, no. 2, pp. 32–36, 2020

# Reproducibility-as-a-Service

## What is reproducibility?

- 3-stage process according to ACM[2]:
  1. Repeatability: Same team executes experiment using same setup
  2. Reproducibility: Different team executes experiment using same setup
  3. Replicability: Different team executes experiment using different setup
- Our testbed-driven approach mainly targets the experimental setup

➔ Focus our effort on repeatability and reproducibility

➔ Replicability requires additional effort by others

[2] [2] ACM, Artifact Review and Badging Ver. 1.1, 2020. [Online]. Available: https://www.acm.org/publications/policies/artifact-review-and-badging-current

# Reproducibility-as-a-Service

## How can we limit effort spent on reproducibility?

- Reduce amount of work for artifact evaluators or other researchers
- Make reproducibility part of experiment design
- → Automate entire experiment (setup, execution, evaluation)

## How can we create robust, reproducible experiments?

- Document all relevant parameters for experiments
- Automate the documentation of experiments
- → Well-structured experiment workflow serving as documentation

# The Plain Orchestrating Service (pos)

## Our solution to create reproducible research

1. Create a testbed management system
2. Create a well-defined experiment workflow

# The Plain Orchestrating Service (pos)

## Our solution to create reproducible research

1. Create a testbed management system
2. Create a well-defined experiment workflow

## Achieving Repeatability

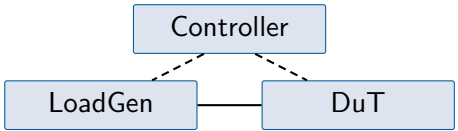
- Automation
- Live images
  - Researchers **must** automate configuration
  - No residual state between reboots

➔ Experiments become repeatable

## Achieving Reproducibility

- Providing access to experiment infrastructure
- Other researchers can easily (re-)run experiment

➔ Experiments become reproducible




Minimal pos experiment topology

# pos' Methodology

## Setup phase

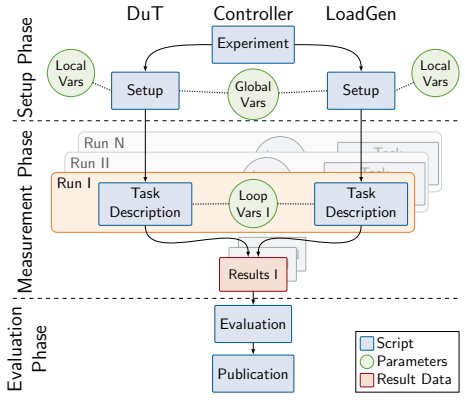
- Controller manages experiment
- Controller configures experiment nodes (DuT, LoadGen)
- Global / local variables (vars) parameterize setup

## Measurement phase

- Repeated execution of measurement script
- Loop variables parameterize each measurement run
  - e.g., different packet rates
  - data of each run is connected to a specific set of loop vars

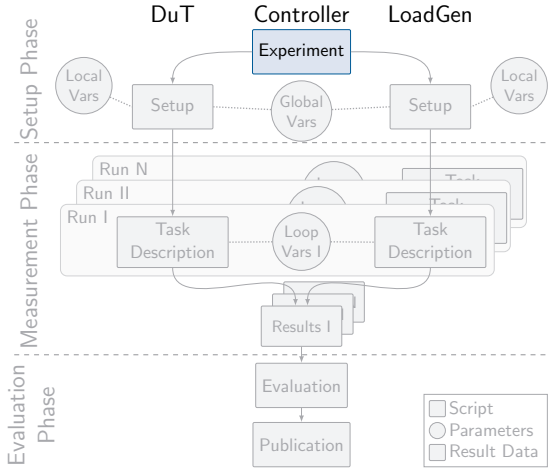
## Evaluation phase

- Collected results / loop vars used for experiment evaluation
- Automated experiment release (git repository, website)




pos workflow

# pos Workflow: Experiment Script




pos workflow

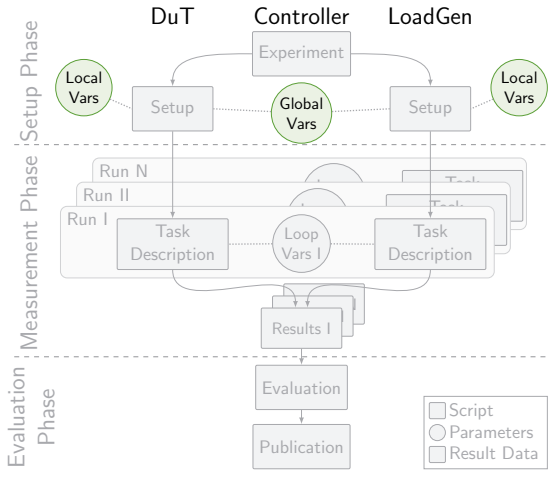
# pos Workflow: Experiment Script

## Experiment Script

- Command line tool: pos
- Specifies the high-level workflow:
  1. Node allocation
  2. Variable loading
  3. Node config and reboot
  4. Node setup
  5. Experiment execution
  6. Node de-allocation

```
# allocate experiment nodes
pos allocations allocate --duration 10 $DUT $LOADGEN

# load variables
pos allocations set_variables $DUT --as-global ./global-variables.yml # global vars
pos allocations set_variables $DUT ./node1/node1.yml                  # local vars
pos allocations set_variables $LOADGEN ./node2/node2.yml              # local vars
pos allocations set_variables $DUT --as-loop ./loop-variables.yml     # loop vars

# configure & reboot nodes
pos nodes image $DUT debian-bullseye
pos nodes image $LOADGEN debian-bullseye
pos nodes reset $DUT --non-blocking
pos nodes reset $LOADGEN --non-blocking

# setup nodes
pos commands launch --infile node1/setup.sh --queued $DUT
pos commands launch --infile node2/setup.sh --queued $LOADGEN

# execute experiment on nodes
pos commands launch --infile node1/measurement.sh --queued --loop $DUT
pos commands launch --infile node2/measurement.sh --blocking --loop $LOADGEN

# free nodes
pos allocations free $DUT
```

# pos Workflow: Global and Local Variables




pos workflow

# pos Workflow: Global and Local Variables

## Variables

- Global variables are available on all participating nodes
- Local variables are available only on specific nodes
- Variables format:
  - YAML
  - JSON

```
repo:
  moongen: 'https://github.com/emmericp/MoonGen'
  commit: '7746ff2f0afdbb222aa9cb220b48355e2d19552b'
```

Listing 1: Global vars

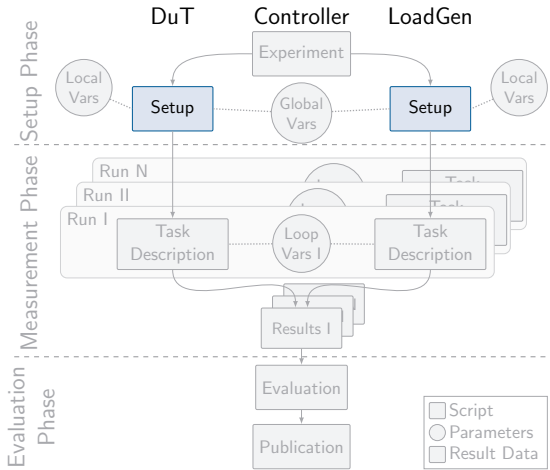
```
interface:
  rx: '1'
  tx: '2'
```

Listing 2: Local vars for LoadGen

```
interface:
  rx: '0'
  tx: '2'
```

Listing 3: Local vars for DuT

# pos Workflow: Setup Script




pos workflow

# pos Workflow: Setup Script

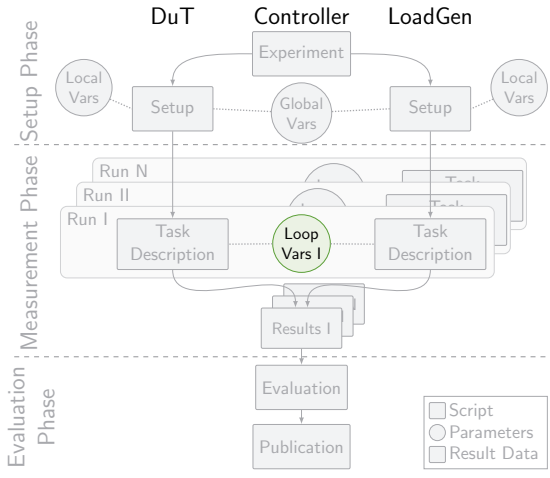
## Setup Script

- Setup script for individual nodes
- Typical high-level workflow:
  1. Get variables
  2. Install dependencies
  3. Configure system
- Format of setup script:
  - Shell scripts for simple setups
  - Ansible to execute complex configuration scripts
  - Any script that can be run on experiment node

```
# get global variables
GIT_REPO=$(pos_get_variable --global repo/moongen)
COMMIT_ID=$(pos_get_variable --global repo/commit)

# clone repo and install MoonGen
git clone --recursive $GIT_REPO /root/moongen
cd /root/moongen
git checkout $COMMIT_ID
./build.sh
./setup-hugetlbfs.sh
```

Listing 4: Setup script for load generator and DuT

# pos Workflow: Loop Variables




pos workflow

# pos Workflow: Loop Variables

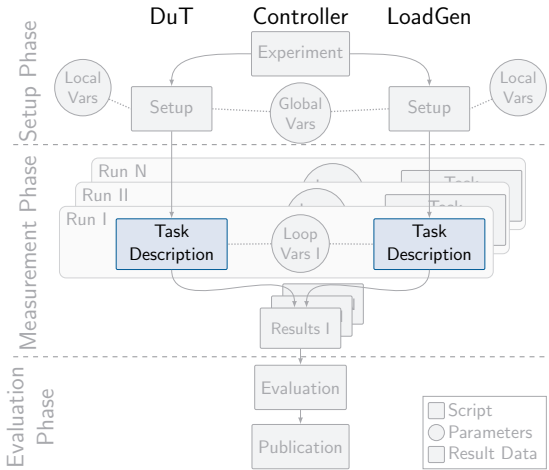
## Loop Variables

- Variables to parameterize the actual measurements
- Typically lists of parameters
- Cross product is automatically created by pos to investigate each possible combination of the specified parameters
- Variables format:
  - YAML
  - JSON

```
pkt_sizes: [
  64,
  128,
  256,
  512,
  768,
  1024,
  1280,
  1518
]
rates: [
  1,
  2,
  ...,
  10
]
```

Listing 5: Loop variables to parameterize the following measurements

# pos Workflow: Task Description




pos workflow

# pos Workflow: Task Description

## Task Description

- pos executes the task description once for each set of parameters
- Typical high-level workflow:
  1. Get (loop) variables
  2. Wait for other nodes (`pos_sync`)
  3. Perform actual measurement
  4. Upload results (`pos_upload`)
- Format of setup script:
  - Shell scripts for simple setups
  - Ansible to execute complex configuration scripts
  - Any script that can be run on experiment node

```
# get global variables
RX=$(pos_get_variable --local interface/rx)
TX=$(pos_get_variable --local interface/tx)
PKT_SZ=$(pos_get_variable --loop pkt_sizes)
RATE=$(pos_get_variable --loop rates)

# wait for other hosts
pos_sync

# start load generator
cd /root/moongen
pos_run --loop loadgen -- \
  MoonGen l3-load-latency.lua -s $PKT_SZ \
  -r $RATE $RX $TX -f rates.csv
sleep 120
pos_kill loadgen

# upload results
pos_upload --loop rates.csv
```

Listing 6: Task script for the load generator

# pos Workflow: Task Description

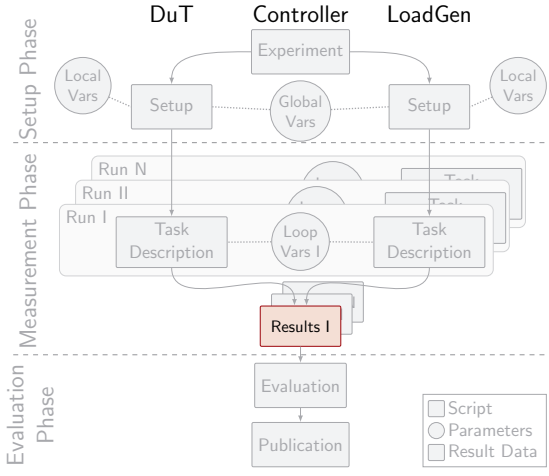
## Task Description

- pos executes the task description once for each set of parameters
- Typical high-level workflow:
  1. Get (loop) variables
  2. Wait for other nodes (`pos_sync`)
  3. Perform actual measurement
  4. Upload results (`pos_upload`)
- Format of setup script:
  - Shell scripts for simple setups
  - Ansible to execute complex configuration scripts
  - Any script that can be run on experiment node

```
# get global variables
RX=$(pos_get_variable --local interface/rx)
TX=$(pos_get_variable --local interface/tx)

#wait for other hosts
pos_sync

# start the forwarder
cd /root/moongen
pos_run --loop fwd -- MoonGen l2-forward.lua $RX $TX
sleep 120
pos_kill fwd
```

Listing 9: Task script for the DuT

# pos Workflow: Result




pos workflow

# pos Workflow: Result

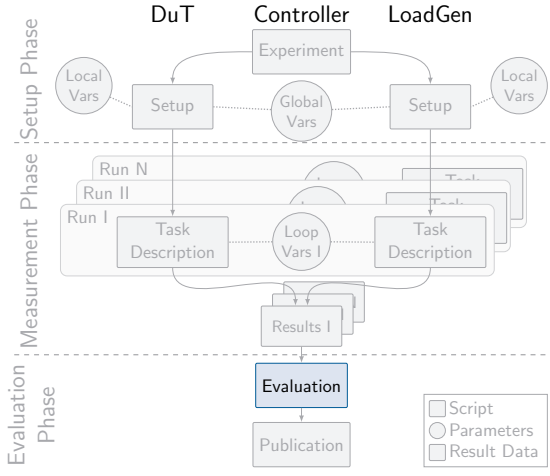
## Result

- Result files are collected on management host
- Result files are annotated with specific loop parameter set
- Format:
  - Command line output
  - CSV
  - Arbitrary formats possible

```
"timestamp";"rx_mpps";"tx_mpps";
1666295702860;0.0;0.0;
1666295703857;0.0;0.0;
1666295704856;3.01;2.97;
1666295705859;12.49;12.47;
1666295706861;12.47;12.48;
1666295707858;12.46;12.45;
1666295708859;12.48;12.49;
1666295709862;12.47;12.48;
```

Listing 10: Example output file containing generated and received throughput

# pos Workflow: Evaluation

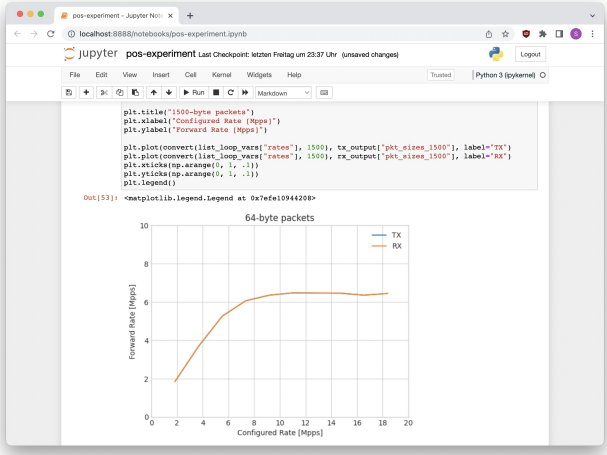



pos workflow

# pos Workflow: Evaluation

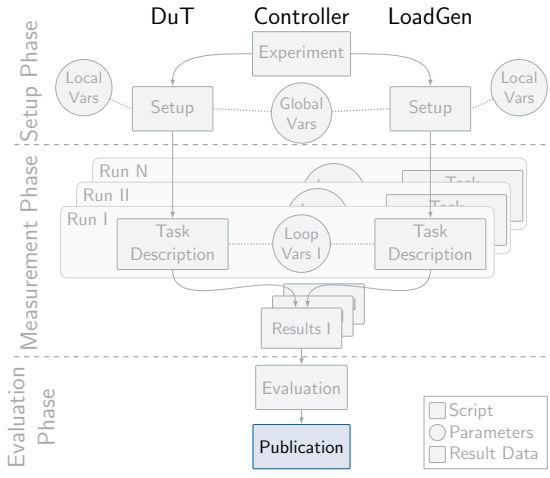
## Evaluation

- Automated evaluation is essential part of experiment design
- We propose to use Jupyter notebooks for evaluation
  - Jupyter supports a variety of different programming languages
  - Jupyter notebooks can be nicely evaluated manually (via browser)
  - Jupyter notebooks can be run on command line automatically




Jupyter notebook evaluation

# pos Workflow: Publication

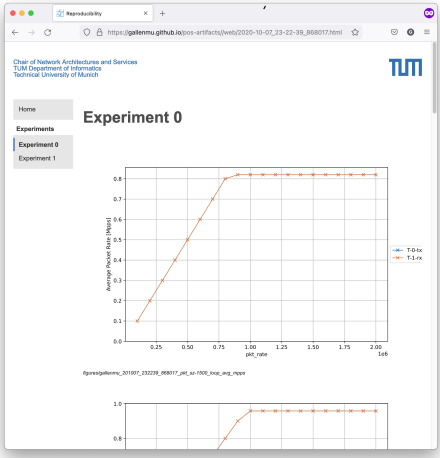



pos workflow

# pos Workflow: Publication

## Publication

- Well-defined workflow:
  - Documentation for people familiar with workflow
  - Release of experiment and results as repository
  - Automated processing of experimental data as website

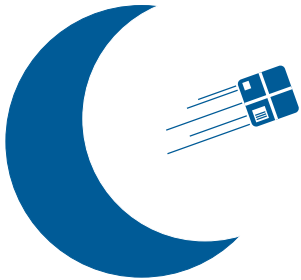



Website generated by pos experiment workflow

# Recommended Tools

- Git repository for scripts
  - Can be used for development of experiments
  - Valuable benefit: keeps track of different versions
- MoonGen for packet generation
  - Any software tool can be installed/used in experiments
  - We rely mostly on MoonGen for generating test traffic
  - Its high precision and accuracy support reproducible experiments
- Ansible for configuration management
  - Setup script can be done in any language
  - For simple setups we typically use shell scripts
  - For complex setups we recommend Ansible
- Jupyter for evaluation
  - Evaluation scripts can be done in any language
  - Jupyter Notebooks are widely used
  - Heavily used by other testbeds (e.g., Chameleon)
  - *Alternative tool* CWL (Common Workflow Language), mainly used in ML/AI domain



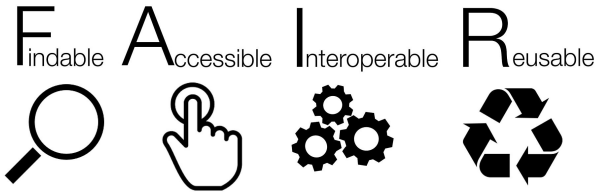



# Data Management Aspects

## Additional aspects to consider

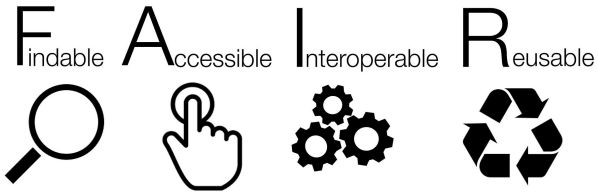
- Reproducibility of experiments is important but not sufficient
- Data management aspects:
  - Experimental artifacts must support the FAIR data principles
  - pos controller offers enough flexibility to employ well-known data and metadata formats
  - pos controller also offers the possibility to automate data collection

## FAIR



# Data Management Aspects

## FAIR




Graphics: Sangya Pundir / CC BY-SA 4.0

- Findable:
  - Automated generation of metadata
  - Structured experimental workflow
- Accessible:
  - Well-known formats (pcaps, JSON, YAML, Jupyter)
  - Well-known tools (git, Gitlab, GitHub)
- Interoperable:
  - Well-known formats (pcaps, JSON, YAML, Jupyter)
  - Well-known tools (git, Gitlab, GitHub)
- Reusable:
  - Structure that documents experiments
  - Repeatable, modifiable and publishable experiments

# Summary

- pos[4] is . . .
  - a testbed orchestration service, and
  - an experiment methodology.
- Methodology makes experiments . . .
  - repeatable as everything is automated,
  - reproducible as others can re-run the automated pos experiments, and
  - easier to replicate as the experiment scripts document experiments.

➔ pos reduces the effort to create reproducible experiments.

➔ pos complements the ACM awards—it does not replace them.

[4] [3] S. Gallenmüller et al., “The pos framework: A methodology and toolchain for reproducible network experiments”, in CoNEXT ’21, Virtual Event, Munich, Germany, December 7 - 10, 2021, ACM, 2021, pp. 259–266. DOI: 10.1145/3485983.3494841

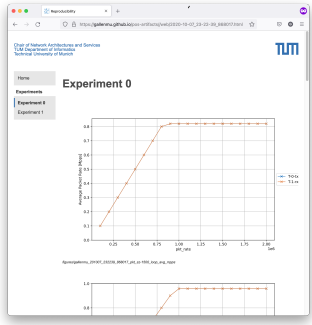
# Summary

- pos[4] is . . .
  - a testbed orchestration service, and
  - an experiment methodology.
- Methodology makes experiments . . .
  - repeatable as everything is automated,
  - reproducible as others can re-run the automated pos experiments, and
  - easier to replicate as the experiment scripts document experiments.

➔ pos reduces the effort to create reproducible experiments.

➔ pos complements the ACM awards—it does not replace them.

- Example experiment:
  - Repository: https://github.com/gallenmu/pos-artifacts
  - Website: https://gallenmu.github.io/pos-artifacts




Website generated by pos experiment workflow

[4] [3] S. Gallenmüller et al., “The pos framework: A methodology and toolchain for reproducible network experiments”, in CoNEXT ’21, Virtual Event, Munich, Germany, December 7 - 10, 2021, ACM, 2021, pp. 259–266. DOI: 10.1145/3485983.3494841

# Mini-Lecture: VLAN
# Virtual Local Area Network

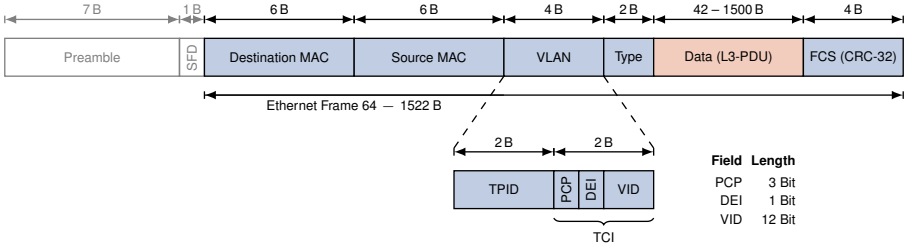
## General Information

- Standardized in IEEE 802.1Q
- Incorporated inside the Ethernet header
- Tunnel endpoints are managed switches
- One physical network to provide multiple separate virtualized Layer 2 networks

## Use cases

- Separate “secure” network from “public” network (e. g. CCTV cams)
- Separate different business units (Development, HR, Finances, . . . )
- Characterize traffic (see QoS)

# Mini-Lecture: VLAN
## Virtual Local Area Network Header Layout




- VLAN header is inserted between source MAC and ethertype
- Ethernet frames having a VLAN header are called tagged (normal frames are called untagged)
- VLAN header consists of 4 fields:
  - TPID: “Tag Protocol Identifier”, always 0x8100, used to indicate that a frame is tagged
  - PCP: “Priority Code Point”, prioritization of traffic, can be used to prioritize different classes of traffic (c.f. IEEE 802.1p)
  - DEI: “Drop Eligible Indicator”, describes if the frame may be dropped in case of congestion
  - VID: “VLAN Identifier”, identifies to which VLAN this frame belongs, from 1 to 4094 (0 and 4095 reserved), most important field

# Mini-Lecture: VLAN
# Access Ports and Trunk Ports

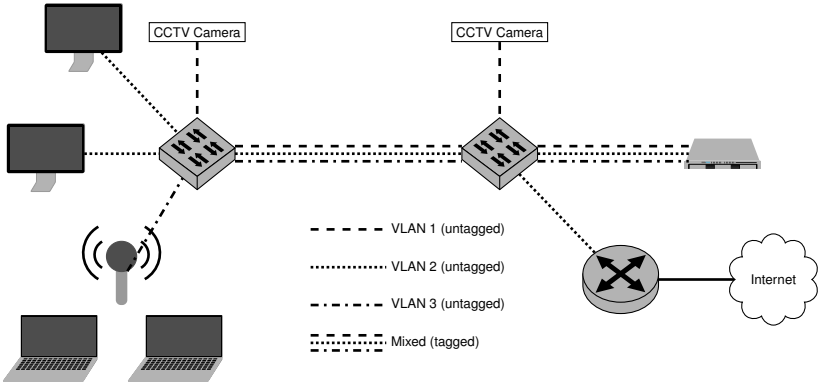
## Access Ports

- Traffic sent to / from this port is not tagged
- Network connected to an access port is logically in one single VLAN
- “The port you connect your desktop to”

## Trunk Ports

- Can send / receive traffic from multiple VLANs
- Tagged frames are forwarded unchanged
- Every untagged frame is tagged using the native VLAN
- Typical switch-to-switch link
- Use with VLAN-aware hosts (VLAN must be configured on the end host)

# Mini-Lecture: VLAN
## Example network




- Switch-to-Switch ports are trunk ports
- Switch-to-Server port is a trunk port
- All other switch ports are access ports

# Mini-Lecture: VLAN
# Q-in-Q (stacked VLANs)

## Encapsulate VLANs in VLANs

- Defined in IEEE 802.1ad
- Two VLAN headers instead of one (Dst MAC | Src MAC | VLAN | VLAN | Ethertype | ... | FCS)
- Total of $4094 \cdot 4094 = 16760836$ VIDs

## Use Case: 4094 VIDs are not sufficient

- Large networks may need more than 4094 VLANs
- Expanding the VID space is enough

## Use Case: Customer network on top of provider network

- ISPs or data centers use one VLAN per customer
- Customer are isolated from each other
- Customers want to use VLANs themselves
- “Lower” VLAN header is managed by the datacenter / provider
- “Upper” VLAN header is managed by the customer

# Mini-Lecture: VXLAN
## Motivation - Virtual eXtensible Local Area Network
### General Information

- Standardized in 2014 in RFC 7348 (rather short standard)
- Builds layer 2 overlay network on top of a layer 4 (UDP) underlay network
- Has 24 bit VXLAN network identifier (VNI), which allows 16 million virtualized networks
- Suitable to reach VMs in large data centers / “the cloud”

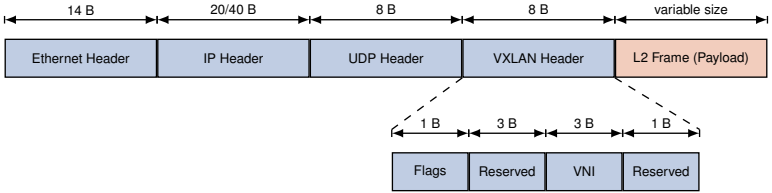
### Problem Statement

- Servers host a large number of VMs
- Each VM has its own MAC address
- VMs need to connect to VMs on other servers
- Switch needs to handle thousands of MAC addresses of VMs

### Another Problem Statement

- Provider and clients both want to use VLANs
- Provider allocates VLANs to clients
- Very limited amount of VLANs per client
- Clients may misconfigure the VMs
- Also solved by Q-in-Q (stacked VLANs), but this is not always applicable

# Mini-Lecture: VXLAN
## Approach




### Encapsulation Strategy

- Encapsulate original layer 2 frame inside UDP
- Virtual networks enumerated by VXLAN Network Identifier (VNI)

### UDP header fields

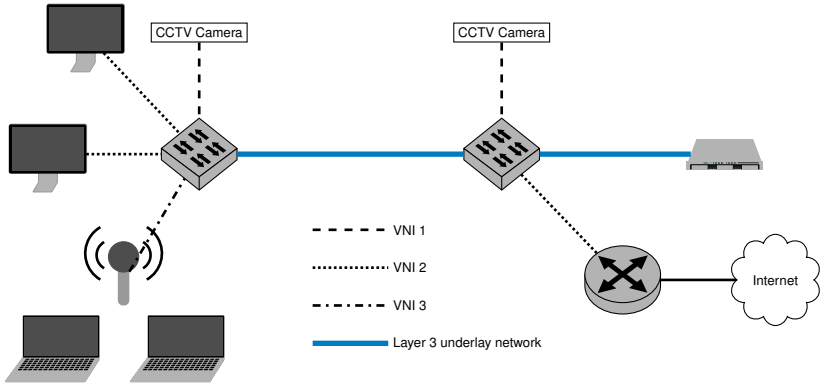
- Source Port: Hash of inner 5-tuple ⟶ great for load balancing
- Destination Port: Always 4789
- Length: Length of layer 2 frame + UDP header size

# Mini-Lecture: VXLAN
## Benefits

### What makes VXLAN a “good” tunneling protocol?

- Builds on top of a layer 3 with only multiplexing on layer 4 (done by UDP)
  - Network may belong to an ISP
  - “The Internet” is layer 3
  - VXLAN can be used over the Internet, VLAN cannot
- Layer 3 routing protocols can be used (BGP, OSPF, . . . )
- Better multipath support

# Mini-Lecture: VXLAN
## Example network



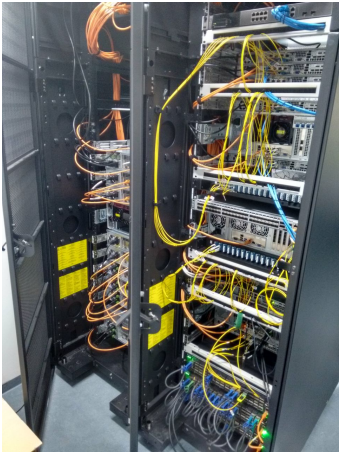

- Links marked as VNI 1/2/3 contain normal Ethernet frames
- Layer 3 network is some arbitrary layer 3 network (e.g. an ISP)
- The two switches encapsulate (/ decapsulate) to (/ from) the VXLAN frames
- Remark: Real world VXLAN-capable switches violate strict layering and use L3 information

# ACN Infrastructure: Motivation

## Hardware testbed

- Hardware typically offers the best performance
- Router project:
  - 4 machines for 12 setups
  - requires 48 hosts in total

➔ Our hardware testbed is not large enough to accommodate all students participating



Hardware testbed

# Suitable Environments for Prototyping Testbeds

| |
|---|
| Emulation [4] |
| Simulation |
| Real hardware |

Requirements and their relative weight

[4] B. Lantz et al., "A Network in a Laptop: Rapid Prototyping for Software-Defined Networks," in ACM Workshop on Hot Topics in Networks. HotNets 2010, Monterey, CA, USA - October 20 - 21, 2010, ACM, 2010, p. 19. [Online]. Available: https://doi.org/10.1145/1868447.1868466

# Suitable Environments for Prototyping Testbeds

| | Flexibility |
|---|---|
| Emulation [4] | + |
| Simulation | + |
| Real hardware | -- |

Requirements and their relative weight

- Flexibility: Network topologies are more restricted on real hardware

[4] B. Lantz et al., "A Network in a Laptop: Rapid Prototyping for Software-Defined Networks," in ACM Workshop on Hot Topics in Networks. HotNets 2010, Monterey, CA, USA - October 20 - 21, 2010, ACM, 2010, p. 19. [Online]. Available: https://doi.org/10.1145/1868447.1868466

# Suitable Environments for Prototyping Testbeds

| | Flexibility | Scalability |
|---|---|---|
| Emulation [4] | + | + |
| Simulation | + | ++ |
| Real hardware | -- | -- |

Requirements and their relative weight

- Flexibility: Network topologies are more restricted on real hardware
- Scalability: Number of nodes and links is typically limited on real hardware

[4] B. Lantz et al., "A Network in a Laptop: Rapid Prototyping for Software-Defined Networks," in ACM Workshop on Hot Topics in Networks. HotNets 2010, Monterey, CA, USA - October 20 - 21, 2010, ACM, 2010, p. 19. [Online]. Available: https://doi.org/10.1145/1868447.1868466

# Suitable Environments for Prototyping Testbeds

| | Flexibility | Scalability | Interactivity |
|---|---|---|---|
| Emulation [4] | + | + | + |
| Simulation | + | ++ | -- |
| Real hardware | -- | -- | + |

Requirements and their relative weight

- Flexibility: Network topologies are more restricted on real hardware
- Scalability: Number of nodes and links is typically limited on real hardware
- Interactivity: Simulation environments do not behave like real systems from a user's perspective

[4] B. Lantz et al., "A Network in a Laptop: Rapid Prototyping for Software-Defined Networks," in ACM Workshop on Hot Topics in Networks. HotNets 2010, Monterey, CA, USA - October 20 - 21, 2010, ACM, 2010, p. 19. [Online]. Available: https://doi.org/10.1145/1868447.1868466

# Suitable Environments for Prototyping Testbeds

| | Flexibility | Scalability | Interactivity | Performance |
|---|---|---|---|---|
| Emulation [4] | + | + | + | -- |
| Simulation | + | ++ | -- | - |
| Real hardware | -- | -- | + | ++ |

Requirements and their relative weight

- Flexibility: Network topologies are more restricted on real hardware
- Scalability: Number of nodes and links is typically limited on real hardware
- Interactivity: Simulation environments do not behave like real systems from a user's perspective
- Performance: Emulation and simulation hardly deliver the performance of real hardware

[4] B. Lantz et al., "A Network in a Laptop: Rapid Prototyping for Software-Defined Networks," in ACM Workshop on Hot Topics in Networks. HotNets 2010, Monterey, CA, USA - October 20 - 21, 2010, ACM, 2010, p. 19. [Online]. Available: https://doi.org/10.1145/1868447.1868466

# Suitable Environments for Prototyping Testbeds

| | Flexibility | Scalability | Interactivity | Performance |
|---|---|---|---|---|
| Emulation [4] | + | + | + | -- |
| Simulation | + | ++ | -- | - |
| Real hardware | -- | -- | + | ++ |

Requirements and their relative weight

- Flexibility: Network topologies are more restricted on real hardware
- Scalability: Number of nodes and links is typically limited on real hardware
- Interactivity: Simulation environments do not behave like real systems from a user's perspective
- Performance: Emulation and simulation hardly deliver the performance of real hardware

➔ Properties of emulation and simulation environments differ significantly from real hardware

➔ We need a tool that better approximates real hardware

[4] B. Lantz et al., "A Network in a Laptop: Rapid Prototyping for Software-Defined Networks," in ACM Workshop on Hot Topics in Networks. HotNets 2010, Monterey, CA, USA - October 20 - 21, 2010, ACM, 2010, p. 19. [Online]. Available: https://doi.org/10.1145/1868447.1868466

# Testbed on a Single System

## Our solution

- We propose a testbed design that relies on virtualization
- → **T**estbed **o**n **a** **s**ingle sys**t**em (toast)

## Advantages

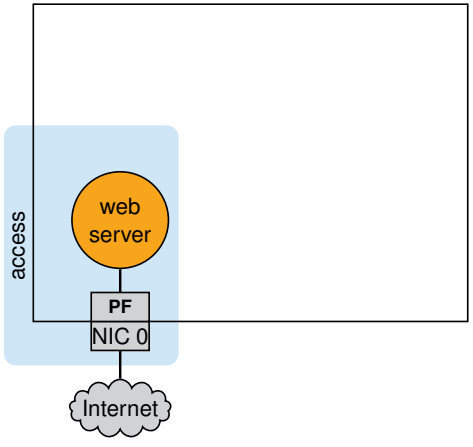
- Interactivity: VMs behave like real systems from a user's perspective
- Performance: VMs can deliver near-native performance (if configured correctly [5])
- Flexibility: Virtual network topologies are highly flexible
- Scalability: Multiple VMs can be hosted on a single server

[5] S. Gallenmüller et al., "5G QoS: Impact of Security Functions on Latency", in NOMS 2020 - IEEE/IFIP Network Operations and Management Symposium, Budapest, Hungary, Apr. 20-24, 2020, IEEE, 2020, pp. 1–9

# Architecture

## Access module

- Access control module separated from testbed implementation
- Possible access methods:
  - Secure Shell (SSH)
  - Webbrowser
- Improved security:
  - Widely used authentication method (OpenID)
  - User data is kept separate from testbed at the authentication service provider
  - OpenID authentication service can be hosted locally or externally
  - External providers available (fast & easy to set up):
    - GitHub, GitLab, Google, etc.

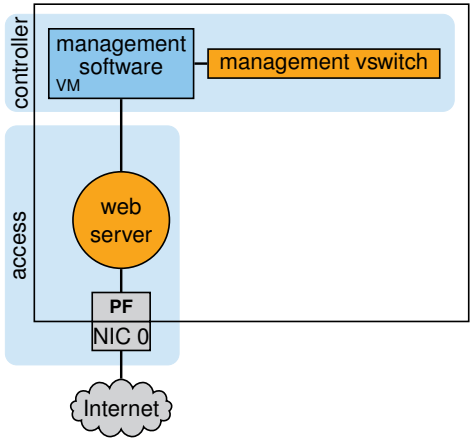



toast architecture overview

# Architecture

## Controller module

- Testbed controller is fully independent from hosting server:
  - Specific OS or configuration for testbed controller becomes possible
  - Testbed controller inside the VM can be changed easily
  - Increased security through additional barrier between host server and virtualized testbed controller
- For toast, we currently use the pos testbed controller [3]



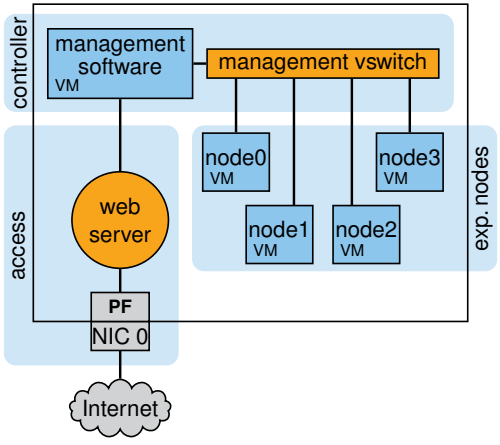

toast architecture overview

[3] S. Gallenmüller et al., "The pos framework: A methodology and toolchain for reproducible network experiments," in CoNEXT '21, Virtual Event, Munich, Germany, December 7 - 10, 2021, ACM, 2021, pp. 259–266. DOI: 10.1145/3485983.3494841

# Architecture

## Experiment nodes module

- Each experiment nodes realized as its own VM
- Virtual switch connects the experiment nodes to the testbed controller
- Virtualization allows high flexibility for experiment nodes:
  - Multiple nodes can be virtualized on a single server
  - Different kinds of nodes can be virtualized depending on the resource allocation:
    - VM resources can be limited to virtualize resource constrained devices
    - High-performance VMs are allowed to allocate more resources

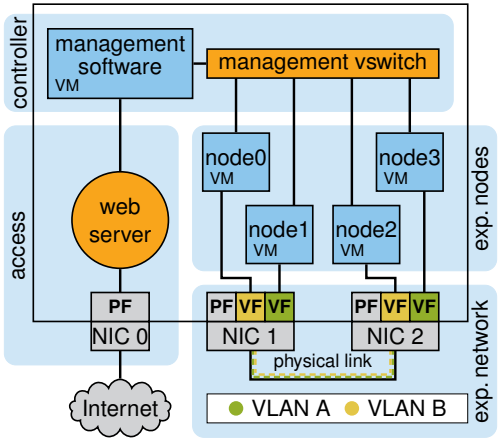



toast architecture overview

# Architecture

## Experiment network module

- Experiment network uses real hardware
- NIC split into virtual NICs via single-root IO virtualization (SR-IOV)
  - Physical function (PF):
    - Bound to host system
    - Manages settings of NIC (bandwidth limits, VLAN settings)
  - Virtual function (VF):
    - Bound to a specific VM
    - Associated with a specific VLAN ID
- VLAN handling is transparent for the VM nodes
- Bandwidth limits and isolation between VLANs are enforced by NIC hardware




toast architecture overview

# Measurements — Setup

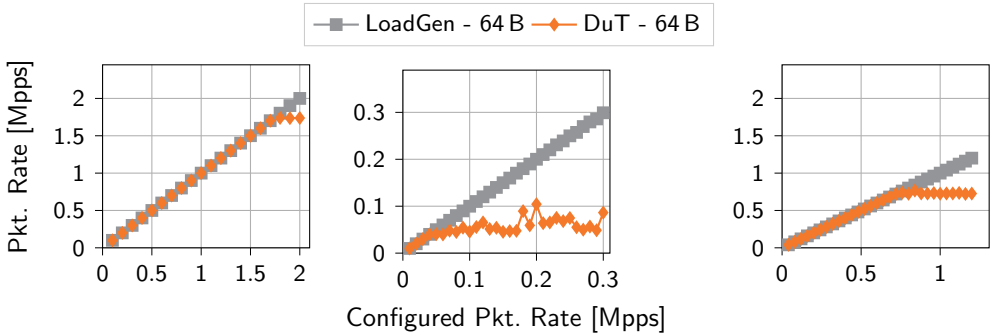

Measurement setup

| | Hardware Testbed / Virtual Testbed | toast Testbed |
|---|---|---|
| CPU | Intel Xeon Silver 4214 (12 cores, 2.2 GHz) | Intel Xeon D-1537 (8 cores, 1.7 GHz) |
| NIC | Intel 82599 | Intel X710-DA4 |
| Linux | Debian buster (Linux kernel v4.19) | Debian bullseye (Linux kernel v5.10) |

## Goal

- Determine the throughput of a Linux router (running on the DuT)
- Direct comparison between three different testbeds:
  - Hardware-based testbed (bare-metal nodes, bare-metal network)
  - Virtualized testbed (virtual nodes, virtual network)
  - Accelerated virtual testbed (toast approach, virtual nodes, SR-IOV-based network)

# Measurements




- Hardware-based testbed delivers highest, most-stable performance
- Virtual testbed delivers lower, comparatively unstable performance
- Accelerated virtual testbed delivers comparatively high and stable performance

→ toast behaves more similar to real testbed than fully virtualized testbed

# Conclusion

## Key takeaways

- Modular architecture makes toast a highly flexible testbed platform
- Hardware acceleration techniques, such as SR-IOV, deliver realistic performance
- toast is not a replacement for a real hardware testbed but a powerful complement:
  - Single-server testbed with low complexity and high affordability
  - Stand-in replacement for a real (future) testbed
  - Development, training, or teaching facility

## ACN server hardware (router project)

- Single-socket AMD EPYC 7763 CPU @ 2.45–3.50 GHz (64 core)
- 1 TB RAM
- Intel E810-CQDA2 (2-port NIC, 100 Gbit/s)

# Bibliography


[1] N. Zilberman, “An Artifact Evaluation of NDP,” Comput. Commun. Rev., vol. 50, no. 2, pp. 32–36, 2020.

[2] ACM, Artifact Review and Badging Ver. 1.1, 2020. [Online]. Available: https://www.acm.org/publications/policies/artifact-review-and-badging-current.

[3] S. Gallenmüller, D. Scholz, H. Stubbe, and G. Carle, “The pos framework: A methodology and toolchain for reproducible network experiments,” in CoNEXT ’21, Virtual Event, Munich, Germany, December 7 - 10, 2021, ACM, 2021, pp. 259–266. DOI: 10.1145/3485983.3494841.

[4] B. Lantz, B. Heller, and N. McKeown, “A Network in a Laptop: Rapid Prototyping for Software-Defined Networks,” in ACM Workshop on Hot Topics in Networks. HotNets 2010, Monterey, CA, USA - October 20 - 21, 2010, ACM, 2010, p. 19. [Online]. Available: https://doi.org/10.1145/1868447.1868466.

[5] S. Gallenmüller, J. Naab, I. Adam, and G. Carle, “5G QoS: Impact of Security Functions on Latency,” in NOMS 2020 - IEEE/IFIP Network Operations and Management Symposium, Budapest, Hungary, Apr. 20-24, 2020, IEEE, 2020, pp. 1–9.