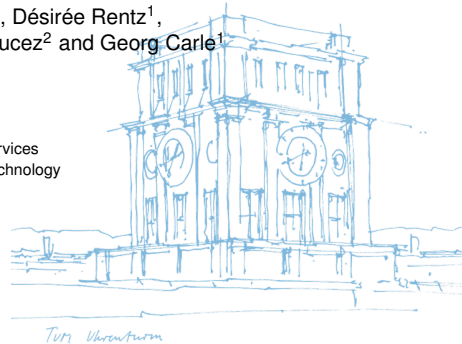


Reproducible Experiment Workflows: The SLICES-pos Testbed Framework

Sebastian Gallenmüller¹, Ziyad Mabrouk², Désirée Rentz¹,
Josef Schönberger¹, Michael Haden¹, Damien Saucez² and Georg Carle¹

¹Chair of Network Architectures and Services
School of Computation, Information and Technology
Technical University of Munich

²INRIA
Sophia Antipolis



Reproducible experiments

- Everyone agrees that reproducible research is important
- The best solution our community has come up so far:

Reproducible experiments

- Everyone agrees that reproducible research is important
- The best solution our community has come up so far:



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. '20¹)



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

What is reproducibility?

- 3-stage process according to ACM²:
 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>

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

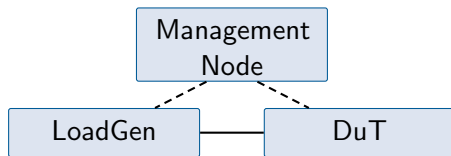
Achieving Repeatability

- Automation
- Stateless testbed images
 - Researchers **must** automate configuration
 - No residual state between resets

→ Experiments become **repeatable**

Achieving Reproducibility

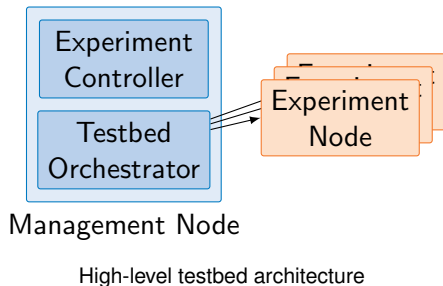
- Providing access to experiment infrastructure
 - Other researchers can easily (re-)run experiment
- Experiments become **reproducible**



Minimal pos experiment topology

Testbed Orchestrator

- Preparation of experiment nodes
- Providing basic infrastructure
- Managed via `slices-bi` command

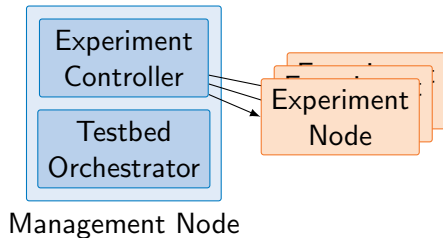


Testbed Orchestrator

- Preparation of experiment nodes
- Providing basic infrastructure
- Managed via `slices-bi` command

Experiment Controller

- Execution of experiments
- Takes over prepared experiment nodes from orchestrator
- Managed via `slices-pos` command



High-level testbed architecture

Setup phase

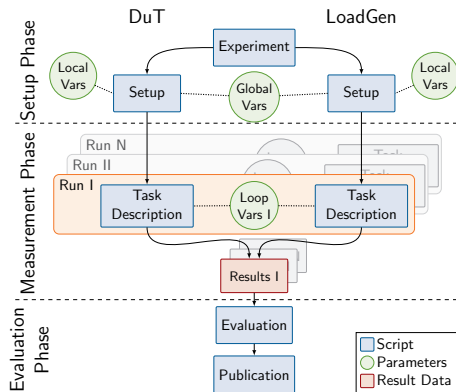
- Controller manages experiment
- Controller configures experiment nodes (DuT, LoadGen)
- Global / local variables (vars) parameterize setup

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)



SLICES-pos experiment workflow

Prepare resources

- Authentication
- Select experiment name
- Select testbed
- Create VMs on selected testbed

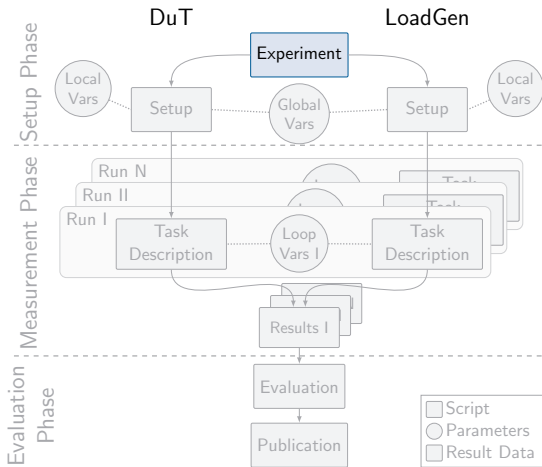
```
# login
slices auth login

# basic infrastructure
export SLICES_EXPERIMENT="pos-experiment"
export SLICES_BI_INFRA="fr-sophia2-bi-vm1"

# create VMs
slices bi create node1
slices bi create node2 --wait

# check bi resources
slices bi list-resources

# check resources in pos
slices pos nodes list
```



Experiment Script

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

```
# allocate experiment nodes
slices pos allocations allocate --duration 10 $NODE1 $NODE2

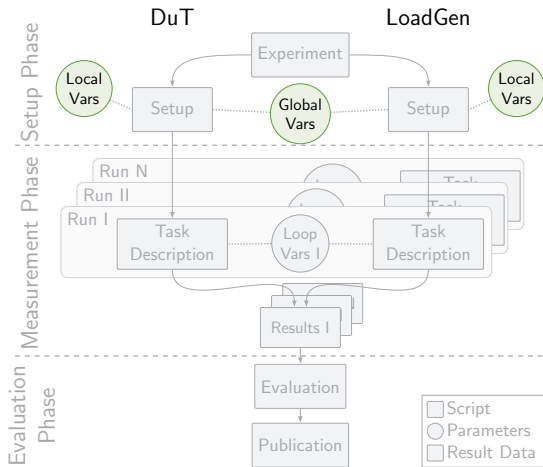
# load variables
slices pos allocations set_variables $NODE1 --as-global experiment/global-variables.yml
slices pos allocations set_variables $NODE1 experiment/node1/node1.yml
slices pos allocations set_variables $NODE2 experiment/node2/node2.yml
slices pos allocations set_variables $DUT --as-loop experiment/loop-variables.yml

# bootstrap
slices pos nodes bootstrap $NODE1 $NODE2

# setup nodes
slices pos commands launch --infile experiment/node1/setup.sh --queued $NODE1
slices pos commands launch --infile experiment/node2/setup.sh --queued $NODE2

# execute experiment on nodes
slices pos commands launch --infile experiment/node1/measurement.sh \
    --queued --loop $NODE1
slices pos commands launch --infile experiment/node2/measurement.sh \
    --blocking --loop $NODE2

# free nodes
slices pos allocations free $NODE1
```



SLICES-pos experiment workflow

Variables

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

```
dataset:  
  name: pos-network-latency  
  output_dir: /tmp/pos-network-latency  
  csv_header: timestamp,ping_count,packet_size, \  
              sender,receiver,avg_rtt_ms
```

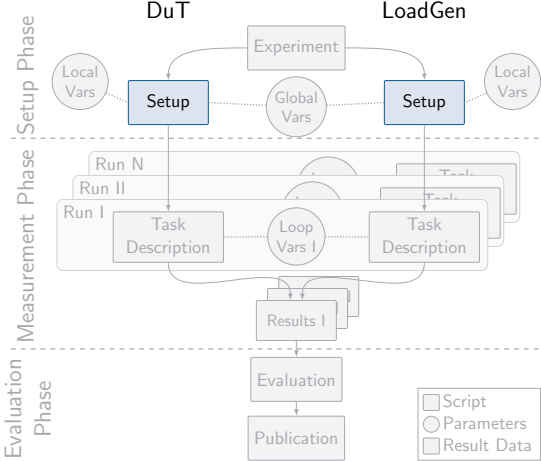
Listing 1: Global vars

```
sender: 10.123.179.23  
receiver: 10.123.210.105
```

Listing 2: Local vars for LoadGen

```
sender: 10.123.210.105  
receiver: 10.123.179.23
```

Listing 3: Local vars for DuT



SLICES-pos experiment workflow

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

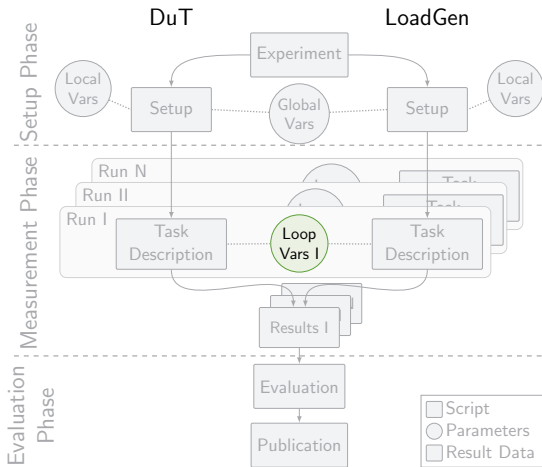
```
# get global variables
name=$(pos_get_variable dataset/name --from-global)
output_dir=$(pos_get_variable dataset/output_dir \
                --from-global)
header=$(pos_get_variable dataset/csv_header \
                --from-global)

# create output directory if missing
mkdir -p "$output_dir"

# create CSV file with header if missing
file="$output_dir/${name}.csv"
if [ ! -f "$file" ]; then
    echo "$header" > "$file"
fi

echo "node1 setup complete: $file"
```

Listing 4: Setup script for Node 1



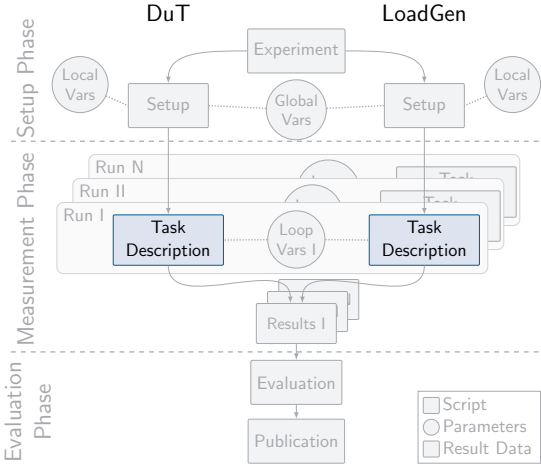
SLICES-pos experiment workflow

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

```
ping_count: [  
    1,  
    5,  
    10  
]  
packet_size: [  
    18,  
    64,  
    256,  
    512,  
    1024  
]
```

Listing 5: Loop variables to parameterize the following measurements



SLICES-pos experiment workflow

Task Description

- pos executes the task description once for each set of parameters
- Typical high-level workflow:
 1. Get variables
 2. Wait for other nodes (pos_sync)
 3. Perform actual measurement
 4. Save results
- Shell scripts for simple experiments
- Different scripts/languages possible

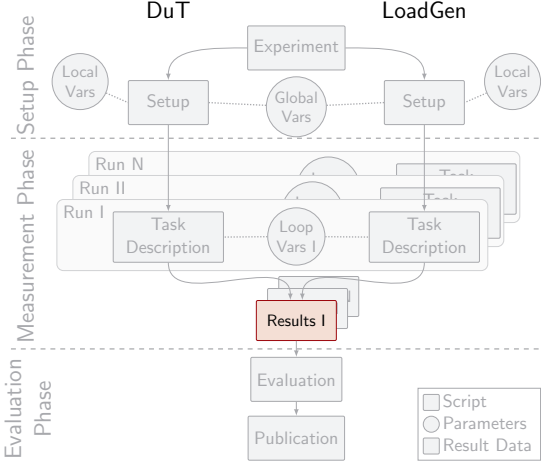
```
# get variables
dataset=$(pos_get_variable dataset/name --from-global)
ping_count=$(pos_get_variable ping_count --from-loop)
packet_size=$(pos_get_variable packet_size --from-loop)
sender=$(pos_get_variable sender)
receiver=$(pos_get_variable receiver)
output_dir=$(pos_get_variable dataset/output_dir --from-global)

# wait for other node
pos_sync
timestamp=$(date -u +"%Y-%m-%dT%H:%M:%SZ")

# run ping and extract average RTT (ms)
avg=""
if ping -c "$ping_count" -s "$packet_size" -I "$sender" "$receiver" | \
tee /tmp/ping_out; then
    avg=$(awk -F"/" ' /rtt/ {print $5; exit}' /tmp/ping_out || true)
fi

# save results
echo "$timestamp,$ping_count,$packet_size,$sender,$receiver,$avg" >> \
"$output_dir/${dataset}.csv"
```

Listing 6: Task script for the load generator



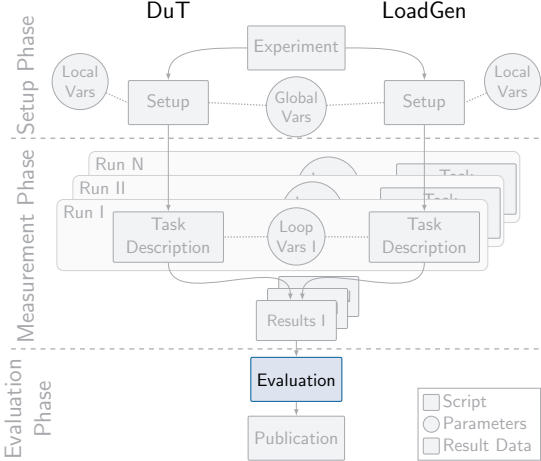
SLICES-pos experiment workflow

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,ping_count,packet_size,sender,receiver,avg_rtt_  
1666295702860,1,18,10.123.179.23,10.123.210.105,1.0  
1666295703857,5,18,10.123.179.23,10.123.210.105,0.9  
1666295704856,10,18,10.123.179.23,10.123.210.105,1.1
```

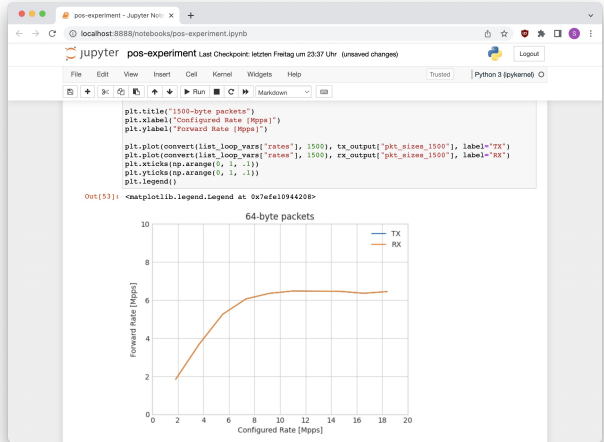
[Listing 7](#): Example output file containing measurement data



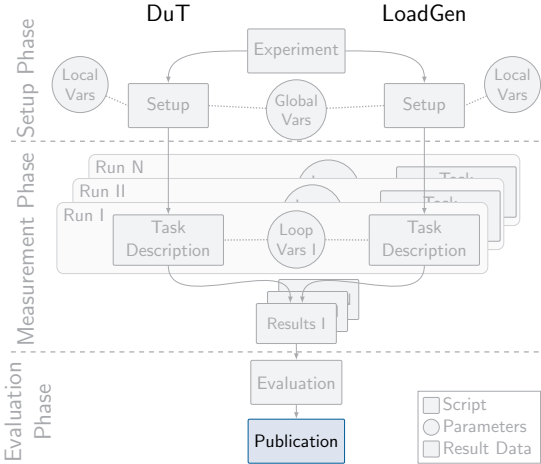
SLICES-pos experiment workflow

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



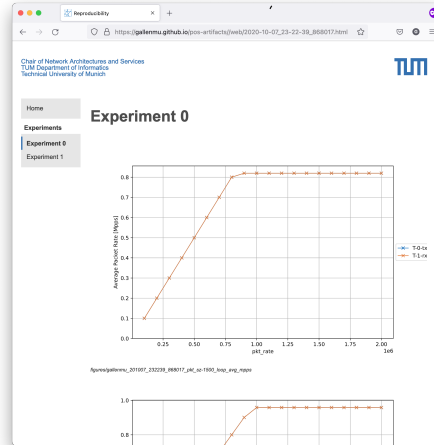
Jupyter notebook evaluation



SLICES-pos experiment workflow

Publication

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

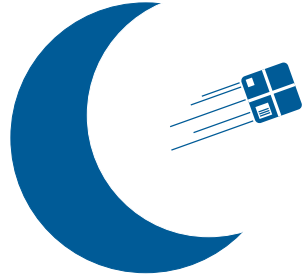


Website generated by pos experiment workflow

- 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



Jason Long / CC BY 3.0



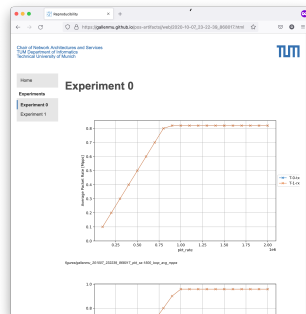
Additional aspects to consider

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

- pos⁴ 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, D. Scholz, H. Stubbe, 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](https://doi.org/10.1145/3485983.3494841)

- pos⁴ 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.



Website generated by pos experiment workflow

⁴[3] S. Gallenmüller, D. Scholz, H. Stubbe, 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

- [1] N. Zilberman, “An Artifact Evaluation of NDP,” *Comput. Commun. Rev.*, Jg. 50, Nr. 2, S. 32–36, 2020.
- [2] ACM, Artifact Review and Badging Ver. 1.1, 2020. Adresse:
<https://www.acm.org/publications/policies/artifact-review-and-badging-current>.
- [3] S. Gallenmüller, D. Scholz, H. Stubbe und 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, S. 259–266. DOI: 10.1145/3485983.3494841.