Reproducible by Design: Network Experiments with pos

Sebastian Gallenmüller, Dominik Scholz, Henning Stubbe, Eric Hauser, Georg Carle
Department of Informatics, Technical University of Munich
Garching near Munich, Germany
{gallenmu, scholz, stubbe, hauser, carle}@net.in.tum.de

***Abstract*—In scientific research, the independent reproduction of experiments is *the* source of trust. Detailed documentation is required to enable experiment reproduction. Reproducibility awards were created to honor the increased documentation effort. In this work, we propose a novel approach toward reproducible research—a structured experimental workflow that allows the creation of reproducible experiments without requiring additional efforts of the researcher. Moreover, we present our own testbed and toolchain, namely, plain orchestrating service (pos), which enables the creation of such experimental workflows. The experiment is documented by our proposed, fully scripted experiment structure. In addition, pos provides scripts enabling the automation of the bundling and release of all experimental artifacts. We provide an interactive environment where pos experiments can be executed and reproduced, available at https://gallenmu.github.io/single-server-experiment.**

***Index Terms*—Reproducibility, Testbed, Network Experiments, plain orchestrating service, pos**

This paper is based on previous work [1], [2].

I. Introduction

Recreating the experimental results of others is a time-consuming endeavor, often lacking novelty and promising low rewards for scientists. Though, reproduction is necessary to verify the original experiment and increase the trust in measured results. Therefore, many scientists agree with the benefits of reproducibility and its incentivization owing to the increased effort in the preparation and release of artifacts for reproduction [3]–[5]. To foster the recreation of experimental results, the ACM [6] introduced badges—awards for papers that invest the additional time and effort to make their results reproducible. We aim to approach the problem from a different perspective: reduce the amount of work researchers have to put into making their experiments reproducible.

We present a methodology for network experiments that relies on an automated experimental workflow. We implement this methodology in our testbed and software toolchain, called the plain orchestrating service (pos). Our testbed depends on scripted experiments to document and automate the experimental workflow to enable the experiments to be repeatable. Our methodology further introduces an experimental structure that divides the experiments into setup, measurement, and evaluation phases, in addition to the division between experiment scripts and experiment parameters. This structure facilitates the documentation of experiments that can be easily shared with others to achieve reproducibility, which requires little additional effort. Using the structured experimental approach and our toolchain, the experiment can be prepared for publication to enable others to replicate the experiments.

We present a methodology that creates inherently reproducible experiments and establish experimental designs that support the researcher to create publishable artifacts with little additional effort. We further provide example pos experiments, including the experimental results, the scripts that enable their creation, and the evaluation. We further provide pos as a platform for others to perform experiments. Therefore, we created a virtual instance of our testbed, demonstrating that the provided artifacts can be easily reused by others and easily created and published by experiment creators.

The remainder of this paper is structured as follows. Sec. II investigates the state of the art for reproducible network experiments. We present our requirements and experimental methodology in Sec. III. In Sec. IV we provide instructions for researchers to use pos. Sec. V concludes the paper.

II. Related Work

Here, we discuss recent developments focusing on tools and testbeds allowing the creation of reproducible experiments.

Reproducibility. In 2015, Collberg and Proebsting [5] studied the replicability of computer science publications. They concluded that the lack of replicability is considered to be a sociological problem, as little reward can be gained from replication. However, more recent developments have suggested that reproducibility is gaining considerable attention through dedicated workshops [7]–[9]. A case study by Zilberman [10] demonstrates that achieving reproducibility is hard, i.e., small variation from the original input, such as the investigated packet size, could lead to a significantly different performance.

Testbeds. Currently, there are numerous initiatives for maintaining and providing testbeds for distributed computing and networking research, e.g., [11]–[16]. The included testbeds specialize in different areas, such as wired networks, 5G, Internet of Things, or Internet-scale measurements. Here, we consider testbeds that target a domain similar to pos, i.e., wired networks and hosts that scale across multiple racks and sites. Nussbaum [17] investigated testbeds [18]–[20] that are similar to pos. He attests to them the ability to execute reproducible network experiments. However, the testbeds neither guarantee nor enforce the creation of reproducible experiments.

Methodologies. In contrast to testbeds, methodologies introduce concepts to structure and execute experiments.

OMF [21] is a testbed controller with its own domain-specific language (DSL) to program network experiments. Similar to pos, OMF enables the creation of reproducible experiments relying on automation. Peuster et al. [22] propose SNDZoo, a repository for reproducible network experiments and a toolchain to reproduce them. Their tools and experiments are focused on containers and VMs. Thanks to the tight integration of methodology and testbed, pos additionally supports low-level hardware experiments. In 2011, Quereilhac et al. [23] presented NEPI, a network experimentation framework that supports various backends such as PlanetLab, netns, or ns-3. pos' methodology goes a step further in experiment design than NEPI, considering subsequent experiment steps such as the evaluation and later publication of data.

pos consists of two entities—its methodology and a testbed implementing this methodology, both conjointly designed with reproducibility in mind. While other testbeds only offer the possibility of creating repeatable experiments, the pos methodology enforces repeatability. Given the access to a pos-capable testbed and experiment files, others can reproduce the experiments, a property that we call reproducibility by design. By following the experimental workflow of pos, reproducible experiments can be created with the requirement of little additional effort. pos cannot ensure replicability; however, the created experimental artifacts document the experiments to enable other researchers to replicate such experiments easily. The complete automation of the experimental workflow further attempts to address the issue of low robustness.

III. Experiment Methodology

The following analysis introduces the requirements for reproducible network experiments: **Heterogeneity (R1)** supports a wide range of different devices such as resource-constrained computers or high-performance servers; **Isolation (R2)** provides means to isolate the experimental network from non-investigated devices; **Recoverability (R3)** reverts the devices into a working, well-defined state in case of misconfiguration or errors; **Automation (R4)** avoids errors or misconfigurations typical for manual setups; **Publishability (R5)** documents the entire experiment automatically to provide others with the necessary information to replicate experimental results. We are not aware of a testbed that adheres to a methodology meeting all identified requirements of pos.

A. Testbed Architecture

We define the *network experiment* as the entire process that configures, performs, evaluates, and optionally publishes measurements. Our network experiments are parameterized with *variables* or *vars*. We call the execution of one concrete instance *measurement run* or short *run* and its outcome *result*.

The high-level architecture of our testbed consists of three communicating roles: one testbed controller managing the experimental workflow and two experiment hosts running the experiment. One of the experiment hosts is the Device under Test (DuT), the object of the investigation for experiments and the other host is the load generator (LoadGen), generating the traffic sent to the DuT. The number of experiment hosts can be scaled. Here, we focus on a minimal topology.

B. Testbed Implementation

To support the different experiment devices (R1), pos implements two APIs: an initialization and a configuration interface. Devices can be initialized using standardized APIs to reset and boot servers such as IPMI, Intel's vPro or AMD's Pro features, or a remotely switchable power plug. The mentioned APIs allow out of band management (R3), i.e., the devices can be reinitialized in the case of configuration errors. After initialization, the configuration interface is used to configure the device and execute the experiment. For a typical Linux server, we use SSH as the configuration interface. Both APIs are extensible to simplify the integration of new devices into pos. The entire initialization process and configuration of a network device is automated via user-defined scripts (R4).

To avoid any shared state between the different executions of the experiment, pos relies on live-boot images. Such images enforce repeatability, as the OS repeatedly starts from a well-defined state, and the researcher must automate and document the device configuration (R3 and R4).

C. pos Experimental Structure

To achieve replicability, other researchers must understand the experimental artifacts. Therefore, we enforce a specific structure to program pos experiments. The user-programmable experiment scripts distinguish two different file types: script and parameter files. In the pos experiment structure, the scripts define the individual steps of the experiment, and the *vars* define the concrete instance of a *run*. For instance, a script file defines the initialization of a network port with the name `$PORT`; the variable file assigns `$PORT` the value `eno1`. This separation allows experiment execution in a different setup by adapting the *vars* without changing the script files.

To further elucidate the experimental structure, we used different scripts for the different participating experiment hosts and the different phases of an experiment. Each experiment host requires two exclusive script files: *setup*, which defines the experiment host configuration, and *measurement*, which defines the active phase of a *run* that generates results. Thereby, a script can be any executable, e.g., python or bash, executed on the target device.

To parameterize the experiment scripts, pos provides three different kinds of *vars*, depending on which experiment host has access and where the *vars* are utilized in the experimental workflow: *global vars*, accessible from all experiment hosts; *local vars*, defined for each experiment host; and *loop vars*, shared across all experiment hosts, but continuously changed between different *runs*.

D. pos Experimental Workflow

In Fig. 1, the script, variable, and result files that describe the high-level workflow of an experiment are presented. Code examples for each file are available [24]. From top to bottom, the workflow is separated into the three subsequent phases:

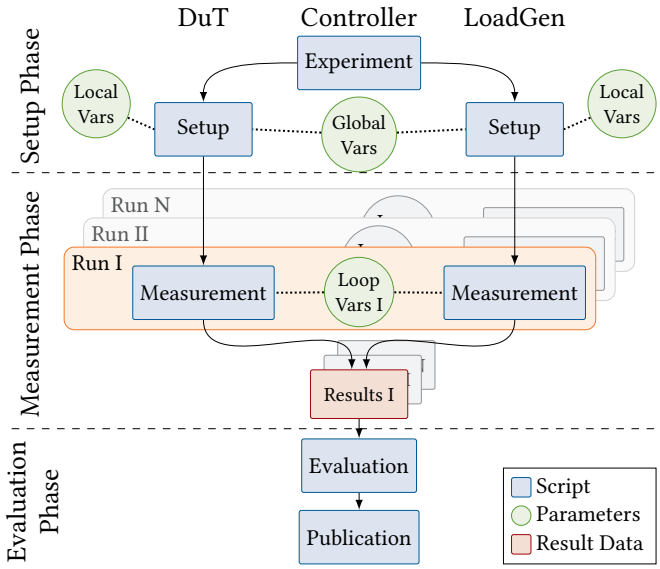

Fig. 1: Experimental workflow

Setup Phase. The testbed controller host executes the main *experiment* script that interacts with the pos API to execute multiple actions. First, it allocates the desired devices, e.g., the DuT and LoadGen. The testbed uses a calendar-based reservation system for the experiment hosts. Devices are configured by loading the *global*, *local*, and *loop vars*. Moreover, a live image and boot parameters can be set for every device. The experiment script instructs pos to start the devices, whereby the boot is executed using the initialization interface. Once the experiment hosts have finished booting, pos deploys a set of utility tools before the setup scripts can be loaded and executed to complete the setup phase. These tools can be used in the setup or measurement scripts; read or set *vars* and synchronize the different experiment hosts. For the configuration of the experiment nodes, pos allows a high degree of flexibility. Users can either script the configuration themselves or use configuration management systems such as Ansible [25], Chef [26], or Puppet [27].

Measurement Phase. During the *runs*, pos executes the measurement script for every host. The number of executions depends on the number of individual parameters in the *loop vars* file. pos experiments perform measurements for each possible combination of loop parameters. If lists are used as parameters, pos automatically generates the cross product over all parameter values. For every set of values contained in the calculated cross product, it executes the measurement script once. The complete output of the experiment script is captured and stored in the result folder of the experiment. This enforced central collection of artifacts, including the utility tools output, executed scripts, *vars*, device hardware, and topology information, guarantees publishability (R5).

Evaluation Phase. The evaluation script typically processes the result files after all runs have been completed. For each *run*, pos creates separate result files and metadata, containing the loop parameters. Based on this metadata, the evaluation script can filter or aggregate specific parameters and values. Our plotting scripts can create throughput figures and latency distributions out-of-the-box using a set of different representations, e.g., line plots or histograms. Our structured experimental workflow allows all artifacts linked to an experiment to be connected, i.e., executed scripts, generated results, and created plots. The *publication* script bundles these artifacts into a release format, e.g., archive or repository. In addition, it generates a website and inserts all the collected artifacts documenting the experimental structure in a format that researchers can easily read.

IV. Testbed-s-a-Service

We provide a repository [28] containing the artifacts of a pos experiment (results, plots, experiment and plotting scripts). The well-defined measurement allows us to generate a website describing the experiment automatically. The website of the previously introduced experiment is available on GitHub [24].

In addition to experimental files, we provide a platform for researchers to develop, execute, and reproduce pos experiments. Our Testbed-as-a-Service (TaaS) platform is available as an interactive shell via web browser [29]. The TaaS platform [2] relies on virtualized experiment hosts instead of real hardware. The experiment network between the VMs is based on single-root IO virtualization (SR-IOV). SR-IOV is a hardware acceleration feature of NICs, that provides hardware-native network behavior and performance for virtualized networks. Therefore, the TaaS platform behaves similarly to a real hardware-based pos testbed.

V. Conclusion

Numerous researchers have proposed ways to embed and foster the spirit of reproducibility in our scientific community through different measures, such as badging, awarding of replicability prizes, or merely allowing appendices that explain experiment replication. We propose a fundamentally different approach: we reduce the effort they have to invest in making their experiments reproducible. Despite the different approaches, our methodology does not replace the incentives for replicable research but complements them.

We operate a virtual Testbed-as-a-Service to enable other researchers to try out pos in their browsers. We do not require researchers to set up their own instance of pos. Instead, we provide them browser access to a virtual instance. A significant advantage of our experimental workflow is that the virtualized experiments can be executed on any pos-driven testbed. Scientists can register their experiments to be executed on pos based on real hardware [29].

Acknowledgment

This project has received funding from the European Union's Horizon 2020 research and innovation programme (project SLICES-SC, 101008468), the Bavarian Ministry of Economic Affairs, Regional Development and Energy (project 6G Future Lab Bavaria), and the German Federal Ministry of Education and Research (16KISK001K).

References

[1] S. Gallenmüller, D. Scholz, H. Stubbe, and G. Carle, "The pos Framework: A Methodology and Toolchain for Reproducible Network Experiments," in *CoNEXT '21: The 17th International Conference on emerging Networking EXperiments and Technologies, Virtual Event, Munich, Germany, December 7 - 10, 2021*. ACM, 2021, pp. 259–266. [Online]. Available: https://doi.org/10.1145/3485983.3494841

[2] S. Gallenmüller, E. Hauser, and G. Carle, "Prototyping Prototyping Facilities: Developing and Bootstrapping Testbeds," in *2022 IFIP Networking WKSHPS: SLICES Scientific Instruments to support digital infrastructure science (IFIP Networking 2022 WKSHPS SLICES)*, Catania, Italy, Jun. 2022.

[3] D. Saucez, L. Iannone, and O. Bonaventure, "Evaluating the artifacts of SIGCOMM papers," *CCR*, vol. 49, no. 2, pp. 44–47, 2019.

[4] N. Zilberman and A. W. Moore, "Thoughts about artifact badging," *Comput. Commun. Rev.*, vol. 50, no. 2, pp. 60–63, 2020.

[5] C. S. Collberg and T. A. Proebsting, "Repeatability in Computer Systems Research," *Commun. ACM*, vol. 59, no. 3, pp. 62–69, 2016.

[6] ACM, "Artifact Review and Badging Version 1.1," 2020, last accessed: 2022-05-06. [Online]. Available: https://www.acm.org/publications/policies/artifact-review-and-badging-current

[7] "ACM SIGCOMM Workshop on Models, Methods and Tools for Reproducible Network Research," 2003.

[8] "Reproducibility '17: Reproducibility Workshop," 2017.

[9] V. Bajpai, A. Brunström, A. Feldmann, W. Kellerer, A. Pras, H. Schulzrinne, G. Smaragdakis, M. Wählisch, and K. Wehrle, "The Dagstuhl Beginners Guide to Reproducibility for Experimental Networking Research," *CCR*, vol. 49, no. 1, pp. 24–30, 2019.

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

[11] "Slices-RI," last accessed: 2022-05-31. [Online]. Available: https://slices-ri.eu/

[12] "Fed4Fire," last accessed: 2022-05-31. [Online]. Available: https://www.fed4fire.eu

[13] "OneLab," last accessed: 2022-05-31. [Online]. Available: https://onelab.eu/

[14] "Grid'5000," last accessed: 2022-05-31. [Online]. Available: https://www.grid5000.fr

[15] "Planetlab," last accessed: 2022-05-31. [Online]. Available: https://www.planet-lab.org/

[16] "Geni," last accessed: 2022-05-31. [Online]. Available: https://www.geni.net/

[17] L. Nussbaum, "Testbeds Support for Reproducible Research," in *Reproducibility Workshop, Reproducibility@SIGCOMM 2017, Los Angeles, CA, USA, Aug. 25, 2017*. ACM, 2017, pp. 24–26.

[18] J. Mambretti, J. H. Chen, and F. Yeh, "Next Generation Clouds, the Chameleon Cloud Testbed, and Software Defined Networking (SDN)," in *2015 International Conference on Cloud Computing Research and Innovation, ICCCRI 2015, Singapore, Oct. 26-27, 2015*. IEEE Computer Society, 2015, pp. 73–79.

[19] R. Ricci, E. Eide, and C. Team, "Introducing cloudlab: Scientific infrastructure for advancing cloud architectures and applications," *login Usenix Mag.*, vol. 39, no. 6, 2014.

[20] D. Balouek, A. Carpen-Amarie, G. Charrier, F. Desprez, E. Jeannot, E. Jeanvoine, A. Lèbre, D. Margery, N. Niclausse, L. Nussbaum, O. Richard, C. Pérez, F. Quesnel, C. Rohr, and L. Sarzyniec, "Adding Virtualization Capabilities to the Grid'5000 Testbed," in *Cloud Computing and Services Science - Second International Conference, CLOSER 2012, Porto, Portugal, Apr. 18-21, 2012*, ser. Communications in Computer and Information Science, vol. 367. Springer, 2012, pp. 3–20.

[21] T. Rakotoarivelo, G. Jourjon, and M. Ott, "Designing and orchestrating reproducible experiments on federated networking testbeds," *Comput. Networks*, vol. 63, pp. 173–187, 2014.

[22] M. Peuster, S. Schneider, and H. Karl, "The Softwarised Network Data Zoo," in *15th International Conference on Network and Service Management, CNSM 2019, Halifax, NS, Canada, Oct. 21-25, 2019*. IEEE, 2019, pp. 1–5.

[23] A. Quereilhac, M. Lacage, C. D. Freire, T. Turletti, and W. Dabbous, "NEPI: An integration framework for Network Experimentation," in *19th International Conference on Software, Telecommunications and Computer Networks, SoftCOM 2011, Split, Croatia, Sept. 15-17, 2011*. IEEE, 2011, pp. 1–5.

[24] S. Gallenmüller, D. Scholz, H. Stubbe, E. Hauser, and G. Carle, "pos Experiment Results and Reproduction (Website)," 2022, last accessed: 2022-05-31. [Online]. Available: https://gallenmu.github.io/single-server-experiment/

[25] R. Hat, "Ansible," 2022, last accessed: 2022-05-31. [Online]. Available: http://ansible.com

[26] Progress, "Progress Chef," 2022, last accessed: 2022-05-31. [Online]. Available: http://chef.io

[27] Perforce, "puppet," 2022, last accessed: 2022-05-31. [Online]. Available: http://puppet.com

[28] S. Gallenmüller, D. Scholz, H. Stubbe, E. Hauser, and G. Carle, "pos Experiment Results and Reproduction (Repository)," 2022, last accessed: 2022-05-31. [Online]. Available: https://github.com/gallenmu/single-server-experiment/

[29] ——, "Virtual Testbed Manager," 2022, last accessed: 2022-05-31. [Online]. Available: https://testtestbed.net.in.tum.de