MARTE: Malleable and Automated Reproduction for Testbed-driven Experiments

Eric Hauser
Technical University of Munich
Garching near Munich, Germany
hauser@net.in.tum.de

Kilian Warmuth
Technical University of Munich
Garching near Munich, Germany
warmuth@net.in.tum.de

Marcel Kempf
Technical University of Munich
Garching near Munich, Germany
kempfm@net.in.tum.de

Stefan Lachnit
Technical University of Munich
Garching near Munich, Germany
lachnit@net.in.tum.de

Georg Carle
Technical University of Munich
Garching near Munich, Germany
carle@net.in.tum.de

Abstract

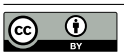
Reproducing experimental results in computing and networking is notoriously difficult due to intricate dependencies and diverse execution environments. Re-executing an experiment often feels like deciphering a puzzle, where incomplete documentation and missing configurations lead to inconsistent outcomes. While research infrastructures in the form of testbeds provide a structured framework for (reproducible) experiments, additional measures are necessary to completely close the loop between experiment execution, sharing of results, and re-execution.

To address these challenges, we propose MARTE, a reproduction methodology for testbed-driven experiments. MARTE consists of two complementary approaches: (1) Malleable reproduction, which retains original scripts, allowing modifications for flexible reuse, and (2) automated record-replay, which automatically captures and seamlessly replays all experiment instructions. Additionally, we integrate a structured data management format to store experiment results along with all relevant metadata about the execution and environment. This new approach for packaging artifacts ensures that configurations and dependencies are preserved and remain accessible. By integrating a tool for seamless publication to open repositories, we transform experimental results into shareable artifacts that others can build upon. We argue that the barriers for researchers to achieve reproducibility must be kept as low as possible. We address this by automating key aspects of experiment documentation, re-execution, and publication.

The functionality of our implementation is showcased in a demonstration experiment. Its complete artifact is openly accessible, along with an example of its reproduction and replay using MARTE.

CCS Concepts

• **General and reference** → **Experimentation**; **Measurement**; *Validation*; • **Networks** → *Network experimentation*; *Network measurement*.



ACM REP '25, Vancouver, BC, Canada

ACM ISBN 979-8-4007-1958-5/2025/07
https://doi.org/10.1145/3736731.3746154

Keywords

Repeatability, Reproducibility, Replicability, Research Infrastructure, Testbed, Experiment, Measurement, RO-Crate, Malleability, Replay

ACM Reference Format:
Eric Hauser, Kilian Warmuth, Marcel Kempf, Stefan Lachnit, and Georg Carle. 2025. MARTE: Malleable and Automated Reproduction for Testbed-driven Experiments. In *ACM Conference on Reproducibility and Replicability (ACM REP '25), July 29–31, 2025, Vancouver, BC, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3736731.3746154

1 Introduction

Reproducibility is a cornerstone of the scientific process, enabling the validation of results and fostering trust in research outcomes. In networking and computer science, achieving reproducibility poses challenges due to the complexity of experimental environments and dependencies on specific hardware and software configurations. Research infrastructures (RIs) give researchers access to large-scale, flexible, and highly configurable testbeds, providing a convenient method for conducting their experiments. In recent years, many RIs like Cloudlab [6], Chameleon [17], or SLICES [8] were established. Although these RIs offer valuable support for conducting reproducible experiments, combining experiment execution, result sharing, and reproduction within a unified and automated framework remains an ongoing challenge that calls for further effort from the research community. Reconstructing experimental environments often remains a manual and error-prone process—highlighting the need for tools that streamline and automate this process.

The challenges of reproducibility are also identified in a study by Collberg et al. [5], who investigate the state of it in more than 600 papers in applied computer science. They evaluate the availability and usability of code and data associated with these publications, finding that a significant number lacked accessible artifacts necessary for re-execution. Moreover, the study identified common challenges to reproducibility: incomplete documentation, missing dependencies, and incompatible environments.

To address these challenges, we present MARTE, a methodology for the malleable and automated reproduction of experiments in testbed-driven environments. The vision of MARTE is to create self-contained experiment artifacts by integrating the results with the scripts that generated them. With MARTE, we want to close the loop between the experiment execution, sharing of artifacts, and

reproduction of experiments. Thereby, MARTE consists of different components.
As the first step, we integrate the RO-Crate [23] structured data packaging approach in the experiment workflow. Using RO-Crate as a foundation, we can store experiment results along with the metadata that describes the entire experimental environment. This integration aims to bridge the gap between experiment execution and documentation, promoting a more holistic approach to reproducible research. Additionally, we store all relevant environment details, i.e. the software and hardware details as well as the network topology. Only by documenting the entire environment, a complete and accurate reproduction of results is possible.
As the main part of this work, we present two complementary approaches to make experiment artifacts fully self-contained. The first approach is a high-level method for the malleable reproduction of experiments, while the second is a low-level method for fully automatic record-replay of experiments.

Malleable Reproduction. This high-level approach is designed to enhance the reusability of experiments by preserving the original scripts that generated the results. Researchers can then arbitrarily edit the experiment scripts in the artifact before initiating the re-execution in an RI. Therefore, this approach promotes the malleability of results, making them shareable artifacts and allowing other researchers to build upon existing work. In this context, malleability means that experiment artifacts can be adjusted, modified, and extended while remaining reproducible. It allows for the complete reproduction of the experiment workflow, from the initial setup through the measurement to the evaluation. However, the implementation of this approach cannot be entirely automatic; it requires minimal effort from researchers to document the exact invocation process. This documentation includes optional arguments and environment variables as well as all additional dependent files of the experiment.

Automated Record-Replay. In contrast, this low-level approach automatically records all instructions issued to the experiment controller during the experiment. This fully automatic approach runs in the background, enabling an effortless and precise replay of the original experiment without manual intervention by the researcher. However, it does not retain the original experiment scripts; instead, it stores the instructions in a machine-readable format. As a result, this method is limited to replay the same experiment without modifications.

The vision of MARTE applies to all three stages of the ACM Artifact Review and Badging [1] terminology: *repeatability*, *reproducibility*, and *replicability*. While *repeatability* and *reproducibility* use the same hardware setup, *replicability* requires a different setup imposing additional challenges. Therefore, ensuring compatibility across RIs is essential for MARTE to address all stages and, thus, promoting the adaptability and transferability of experiments.

The remainder of this paper is structured as follows: Section 2 provides background information on issues in reproducibility, result data management, and related initiatives. Thereafter, we formulate the requirements of our proposed approaches in Section 3. Section 4 introduces the integration of RO-Crate to structure experiment artifacts. The implementation details of the malleable reproduction and automated record-replay approaches are described in Sections 5 and 6, respectively. Section 7 discusses the automated publication and sharing of experiment artifacts, while Section 8 demonstrates our approaches in an example experiment. Finally, Section 9 concludes the paper, summarizing our contributions and outlining potential directions for future research.

2 Background

In this section, we discuss further aspects of reproducibility and provide more background information for our implementation. Moreover, we gather requirements for our implementation as well as explain and justify the tools we use.

2.1 Issues in Reproducibility

As a continuation to the introduction, which presented the study of Collberg et al. [5], several papers describe the challenges of reproducibility but also its importance and value to the community.

Hummel and Manner [13] provide a structured literature review on reproducibility in computer science, highlighting the lack of proper documentation as a major barrier. Their findings emphasize that certain fields, such as Information Retrieval, have made more progress in fostering reproducibility through dedicated efforts and community-driven initiatives. They also note the role of publishers and conferences in promoting reproducibility through artifact evaluation and badging systems.
Gil et al. [10], in their study on scientific workflows, explore the specific challenges faced in reproducing workflow-based research. They identify issues such as missing dependencies, evolving software environments, and inadequate metadata, which frequently hinder successful re-execution. Their work underscores the need for comprehensive provenance tracking and standardization in workflow management to facilitate reproducibility.
Similarly, Mayer and Rauber [21] conducted a quantitative study on the re-executability of publicly shared scientific workflows, revealing that many workflows cannot be easily re-executed due to incomplete or outdated dependencies. Their findings demonstrate that even when workflows are publicly available, a lack of structured and machine-readable metadata can severely impact their usability and reproducibility.

Especially when experiments rely on specialized hardware that cannot be easily replicated, a simple reproduction is difficult. However, we want to close this gap by capturing and providing as much metadata as possible about the experimental environment. Thus, enabling researchers to recreate conditions as closely as possible.

From our understanding, backed up with previously mentioned studies, the barriers and effort required for achieving reproducibility must be as low as possible. Ideally, such a system should work mostly in the background, requiring minimal additional effort from researchers. MARTE addresses this need by automating the reproducibility process to the greatest extent possible. As a positive side effect, experiments that researchers initially considered merely for testing purposes become automatically reproducible.

2.2 Experiment Data Management

The management of experimental data is a crucial aspect of ensuring reproducibility. Since the concept relies on sharing results with

the scientific community, it is essential that these results are accessible to everyone. Moreover, results should provide the relevant context (i.e. metadata) of the experiment.
In 2016, Wilkinson et al. [26] proposed FAIR as guidelines for the organization of research data. FAIR consists of four principles that establish the theoretical foundation for how research data should be organized to make it easily accessible to the community:
Findability: Results should have a globally unique identifier and be indexed in searchable resources.
Accessibility: Data should be retrievable using open and free standard protocols, with metadata remaining accessible even if the dataset itself becomes unavailable.
Interoperability: Results should be presented in widely recognized formats that are machine-readable and should reference related datasets.
Reusability: Datasets should include rich metadata, be licensed appropriately, and follow domain-specific community standards.

As part of the reproducible experiment workflow, online open repositories are important platforms that focus on the findability and accessibility aspects of FAIR. These platforms provide a simple method to publish, preserve, and index data. Different platforms have been established in the past, for example Trovi or Zenodo.
Trovi [4] is an open repository platform for sharing, preservation, and discovery of experiment artifacts. Trovi is mainly used by the Chameleon [17] RI.
Zenodo [7] is another open repository hosted by CERN to publish, share, and preserve research data. Moreover, Zenodo assigns a Digital Object Identifier (DOI) to every artifact, making them uniquely identifiable and, therefore, easily citable. Both, Trovi and Zenodo, provide an integration with GitHub for improved version control. On Zenodo, different versions of the artifact can have individual (sub-)DOIs for clear differentiation. As a result, researchers can push updates and ensure they are easily identifiable.

Regarding MARTE, we tightly connect open repositories to our implementation. Researchers should be able to publish experiment results automatically. Additionally, existing artifacts should be easily downloadable back into the RI to enable seamless reproduction.

Concerning the interoperability and reusability aspects of FAIR, a structured data management format is necessary. One initiative is Research Object Crate (RO-Crate), proposed by Soiland-Reyes et al. [23], which packages research data together with its associated metadata. RO-Crate provides a structured format for describing research outputs and linking related entities together. Thereby, RO-Crate uses a machine-readable format to store all metadata.

2.3 Selected Infrastructure and Tools for MARTE

First, we require an RI as a foundation for the implementation of MARTE. In the following, we list and discuss available RIs.

CloudLab [6] is a federated testbed and RI for cloud computing research and spans over multiple sites across the USA. CloudLab uses Emulab [12] to allocate raw access to different hardware resources like servers or switches. By providing a diverse and reconfigurable infrastructure, it enables researchers to design and test custom cloud environments tailored to their specific experimental needs.

Chameleon [17] is an RI and testbed designed for configurable experiments in edge and cloud-based systems. It offers a flexible platform that provides access to a wide variety of hardware. Instead of using self-developed software, Chameleon utilizes OpenStack and employs a virtual currency for resource allocation. A key feature of the platform is Trovi, which hosts experiments based on Jupyter [19] notebooks, enhancing sharing and reproducibility.

SLICES [8] (Scientific Large-Scale Infrastructure for Computing/ Communication Experimental Studies) is a European RI designed to support large-scale experiments in computing and networking. It provides a federated testbed with advanced computing, storage, and networking resources, enabling researchers to conduct experiments.

We have chosen the SLICES RI for the implementation of MARTE. SLICES utilizes the pos [9] methodology and framework for conducting experiments. One significant advantage of using pos is the compatibility with other RIs. Stubbe et al. [24] demonstrated how experiments from SLICES/pos can be executed on the other well-known RIs Cloudlab [6] and Chameleon [17]. Consequently, the SLICES/pos RI offers the necessary capabilities to attain the highest ACM badge: *replicability*.

2.4 The pos Methodology

Within the SLICES RI, pos [9] is the main methodology and framework for the experiment execution. As a framework, pos manages all experiment machines (i.e. testbed nodes) from a centralized management server running the pos daemon. The management includes several tasks, such as the reservation, allocation, reset, and execution of commands and scripts on the testbed nodes.

As a methodology, the core of pos is the *everything-must-be-scripted* paradigm. All testbed nodes boot live images running in a ramdisk after a reset. As a consequence, a node is always in a well-defined state before an experiment commences. This enforces that the researcher fully scripts the setup and the subsequent measurement, as there is no state-keeping between experiment executions. Moreover, the specification of the used OS image ensures a consistent base environment across executions. While scripting the setup of a node, it is essential for researchers to pin all dependencies to a specific version or commit hash, especially when manually built.

Experiment Procedure. To provide a clearer understanding of a pos experiment—both in general and for the following sections—we briefly outline the procedure in this paragraph.

An experiment starts with the reservation of the available testbed nodes. This happens either directly on the CLI of the management host or in a web-based calendar system. Once the nodes are reserved, the researcher creates an allocation that generates a unique experiment identifier. All subsequent instructions involving the allocated nodes belong to this experiment. After these initial steps, users can send commands to configure the testbed nodes and run scripts directly on them. The configuration includes selecting an operating system image, optionally setting kernel boot parameters, and rebooting the nodes. Moreover, files can be transferred from the management node to the testbed nodes. Next, researchers can execute scripts of any kind on the experiment nodes. Additionally, pos offers the ability to parameterize the scripts using variable sets. These variables can be accessed on the testbed nodes while executing scripts. Moreover, variables can be used in loop mode by

specifying start, step, and stop values. This is particularly useful for experiments that examine how behavior changes across a range of a given input value.

2.5 Related Initiatives

There are very few initiatives that combine testbed-driven experimentation, reproducibility, and artifact sharing. Among them, KheOps by Rosendo et al. [22] is a key initiative in this domain. The authors themselves mention the limited availability of comparable solutions while also emphasizing the potential of such integrated approaches to improve reproducibility in experimental research. KheOps introduces a Jupyter notebook-based environment for describing experiments, integrates the Trovi platform as an open repository for sharing artifacts, and connects different RIs. They also close the loop between experiment execution, artifact publication, and reproduction. Additionally, it promotes collaboration by enabling researchers to reuse and adapt existing experiments.

With MARTE, we address some limitations of KheOps. We provide detailed and automated documentation of the software and hardware setup, including OS images and network topologies. Second, we integrate RO-Crate as a standardized, machine-readable data management format that bundles experimental results together with the scripts that produced them. This ensures complete traceability and avoids undocumented experiments. Additionally, MARTE targets the networking research area more directly, providing details about the network environment, for example optical splitters for non-intrusive traffic measurement.

3 Requirements

For the descriptions of our implementation in the upcoming Sections 4, 5, 6, and 7, we formulate the following requirements. We distinguish between general requirements R-G applicable to both approaches and specific requirements for the individual approaches R-M (malleable reproduction) and R-R (record-replay), respectively. These requirements are later referenced in the respective sections.

3.1 General Requirements

The general requirements define the foundation of MARTE. These requirements refer to the integration of RO-Crate for structuring the results and the subsequent publication of results, which apply to both approaches:

R-G1 *Complete and Structured Metadata Storage*:
All experiment metadata, including hardware and software setups, as well as dependencies and configurations, should be stored in an RO-Crate instance. Results and metadata, as well as the metadata itself, should reference each other to improve the artifact's understandability.

R-G2 *Publication and Sharing of Experiments*:
The implementation must support straightforward publication and sharing processes through standardized formats and metadata. In future, we envision an extensive, categorized collection of available experiments that can be used for reproduction as well as a source of inspiration.

R-G3 *Portability of Experiments*:
The approaches should allow experiment reproduction across different testbeds or environments.

3.2 Malleable Reproduction Requirements

Here, we list requirements of the malleable reproduction approach. The core principles emphasize retaining the original scripts while imposing only minimal restrictions on the execution environment:

R-M1 *Retention of Original Scripts*:
The original experiment scripts must be preserved to avoid any form of information loss.

R-M2 *Malleability of Experiments*:
Researchers should be able to modify existing experiments with minimal effort to explore new ideas.

R-M3 *Minimal Restrictions*:
This approach should impose as few constraints as possible on the experiment execution and modification.

3.3 Record-Replay Requirements

The requirements for the record-replay approach focus on achieving full automation, ensuring that experiments can be recorded and replayed automatically without requiring manual intervention or additional configuration:

R-R1 *Automated Experiment Recording*:
The framework records all instructions issued during the experiment setup, execution, and evaluation. This automated process eliminates the need for manual documentation, reducing human error and ensuring completeness.

R-R2 *One-Click Replay*:
Users must be able to replay experiments through an automated process requiring no manual setup or adjustments.

4 RO-Crate for Experiment Results

Before we can make the artifacts of pos-driven experiments self-contained by combining results with the scripts that produced them, we must first advance the organizational structure of the artifacts. Currently, the experiment results (i.e. artifacts) are stored in an unorganized and unstructured manner, complicating their management, reproduction, and effective sharing. This unorganized structure makes it more difficult to compare different experiments, which is a fundamental aspect of reproducibility. Therefore, we propose restructuring the experiment results of the pos methodology and adopting a formal results management system: RO-Crate [23]. RO-Crate provides a standardized approach for packaging experimental data along with its metadata. The core of RO-Crate is the mandatory `ro-crate-metadata.json` file, located at the root folder of the artifact. This file uses JSON for Linked Data [25] (JSON-LD) to establish references between different artifact components. As a result, the data becomes more interconnected, structured, and easily machine-readable.

In previous work [11], we demonstrated how incorporating additional metadata, such as authors, hardware parameters, and network topology, enhances the completeness of research data.
Building on this foundation, we propose the comprehensive linking of the various single components within the result artifact. Explicitly defining the dependencies within the result artifact makes their relationships more accessible and easier to understand. As a consequence of this approach, we fulfill requirement R-G1.

In the following, we detail our implementation of RO-Crate as a foundation for MARTE. First, we modify the pos daemon to generate

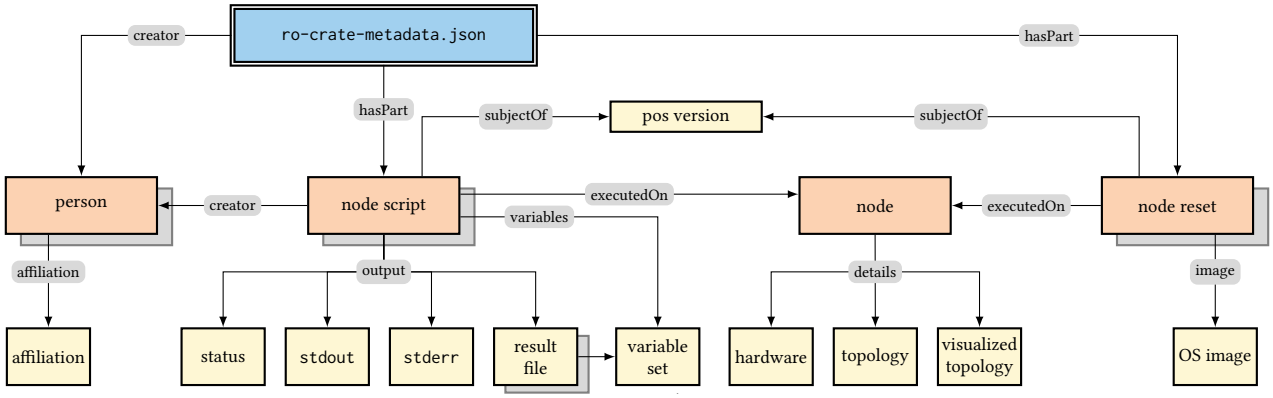


Figure 1: Proposed RO-Crate structure (simplified) for experiment artifacts combining experiment results and metadata.

a new RO-Crate instance for each executed experiment. In such an instance, we include the result data as well as all available metadata. Figure 1 shows the links between the components of an RO-Crate instance. At the top of the figure, there is the previously mentioned `ro-crate-metadata.json` file that stores all metadata information. Since the complete structure of such an instance would be too lengthy to explain in detail, we have simplified it here. Therefore, we focus on three main entities: (1) the involved *persons*, (2) *node resets*, and (3) the *node scripts* of the experiment.

First, an experiment can have multiple involved *persons* that are a part of the experiment. Moreover, every *person* is linked to their institutional affiliation as shown on the left of the figure. We use the Open Researcher and Contributor ID (ORCID) [15] for unambiguously identifying a person and the Research Organization Registry (ROR) [20] for identifying their affiliation.

On the right of the figure, we display the *node resets*. A reset involves rebooting an experiment node and booting a previously selected live OS image into a RAM disk. Each reset entry links to the node on which it was executed and the image used. The used image contains the image's name (for example `debian-bookworm`) as well as a detailed log of installed packages. Currently, we build these images locally from the official Linux distribution's package repositories. To that end, we plan to provide an option to integrate the used images into the RO-Crate artifact. A reset entry also refers to the version of the pos daemon on which the experiment took place. Following the link to the node entry, this entity includes further information about the node. For example, hardware and topology information providing additional context about the experiment's hardware and network setup.

The *node scripts*, located below the `ro-crate-metadata.json` in the figure, refer to scripts that are launched on the experiment nodes. Similar to the *node reset*, a *node script* entity references the node on which the script is launched. Additionally, we connect the set of variables used during execution. This link to the variables helps tracking the specific environment during the execution of a particular *node script*. Furthermore, we link the *node script* to its output, including `stdout`, `stderr`, and `status` (exit code), as well as additional result files it may have produced during execution. Finally, each script is associated with the *person* who initiated the execution.

By adding and linking all these entities, we create a comprehensive and structured RO-Crate instance that packages the complete information about an experiment. Moreover, introducing RO-Crate lays the foundation for self-contained experiment artifacts and enables machine-readable access to the experiment and its results.

The RO-Crate standard also offers the possibility of providing a user-friendly representation of the artifact in the form of a website. This representation allows researchers to understand and navigate the self-contained experiment artifact, including metadata, scripts, and results. We plan to release this visualization concept in a future version of MARTE as it closes the gap between the machine-readable JSON-LD and a human-readable representation.

5 Malleable Reproduction of Experiments

In this section, we present our implementation of the malleable reproduction for testbed-driven experiments. Malleability, in this context, refers to the reusability and adaptability of experiments while ensuring their reproducibility. Specifically, a malleable experiment can be re-executed with modifications, allowing researchers to explore variations in configurations, parameters, and dependencies. As a consequence, to allow modifications, the original scripts need to be retained in a way that also ensures an effortless re-execution. Moreover, the original experiment script may contain additional logic or helpful documentation for other researchers that is worth preserving. However, achieving this form of malleable reproduction requires additional effort from the researcher to precisely document the invocation of the initial experiment script. This includes specifying the execution parameters, environment settings, and dependencies that influence the experiment's outcome. Additionally, all required extra files must be explicitly referenced to ensure that they remain accessible in the artifact. To facilitate this process, we propose a new configuration file format that consolidates the necessary information. This file serves as a structured reference, allowing researchers to:

- Document the exact command-line invocation of the experiment script, including the interpreter and any optional arguments.
- Specify all extra files that are not part of the experiment script's folder.

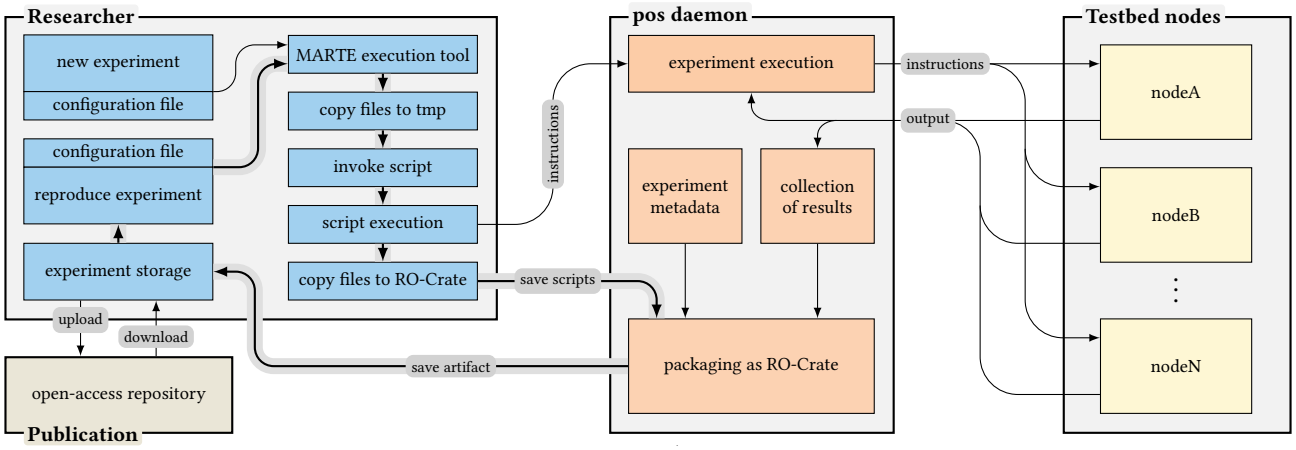


Figure 2: Workflow of the malleable reproduction approach: The main part of this approach occurs on the researcher's side. The closed-loop process is highlighted.

- Set adjustable parameters like environment variables.

By using such a configuration file, we keep the design of the experiment scripts widely open and do not pin on a specific programming or scripting language. The only requirement is that the experiment script is an executable file. This method fulfills the requirement R-M3, focusing on minimal restrictions to the experiment design. As a result, the configuration file offers a structured way to document how to (re-)execute an experiment in the testbed environment. By providing this structured information, we introduce a closed-loop process for experiments. Experiments do not end when they finish but rather provide the foundation for subsequent (modified) executions. In the following paragraph, we explain the workflow of the malleable reproduction approach in more detail.

Workflow. Figure 2 illustrates the workflow of the proposed malleable reproduction approach. The process initially begins with researchers defining a new experiment in an executable file as shown in the top left corner of the figure. Before starting the execution, researchers additionally create the aforementioned configuration file, which specifies the invocation details. To start the execution, the researcher passes the configuration file to a new CLI command introduced by MARTE. This new MARTE execution tool then prepares the experiment execution. First, the main executable file and all other files specified in the configuration file are copied to a temporary location. By taking this step, we ensure that all necessary files are present. Moreover, this approach helps to identify and prevent issues with relative and absolute paths. In general, we recommend using relative paths to ensure the experiment runs correctly in different environments. Next, the execution tool invokes the experiment script within the temporary environment based on the information in the configuration file. The invocation and, therefore, the allocation of the involved testbed nodes triggers the creation of a new RO-Crate instance for this experiment. From this point, the invoked script sends instructions to the pos daemon positioned in the center of the figure, for example resetting nodes or launching scripts on the nodes. Once the experiment script completes, all contents from the temporary folder, along with the configuration file, are transferred to the existing RO-Crate instance of the experiment. This process is depicted in the bottom center of the figure where the RO-Crate instance is packaged. At this point, the workflow is complete, and the resulting RO-Crate contains both the experiment results and the original experiment script that produced them, achieving requirement R-M1.

As the final step, the created RO-Crate instance is saved locally on the testbed management host. Researchers have two options: they can either publish the experiment artifact by uploading it to an open repository or initiate the reproduction of the experiment. Details about the publication process and sharing of result artifacts are outlined in Section 7. For reproduction, a researcher can either rerun the original script as-is or modify specific parts before re-executing it. This flexibility for arbitrary modifications ensures compliance with requirement R-M2. Once the same or another researcher initiates the experiment reproduction, the process forms a closed loop, where an existing experiment serves as the foundation for a new experiment run. The thick arrows in the figure illustrate the closed loop.

6 Automated Record-Replay of Experiments

In addition to the malleable reproduction approach, we introduce a record-replay approach, serving a slightly different use case. This approach provides a simple and lightweight method for replaying experiments without requiring additional effort from the researcher. A key advantage is that, once initiated by the user, the replay runs completely automatically on the pos daemon, requiring no further user interaction. Unlike the malleable reproduction approach, the record-replay method does not retain the original experiment scripts. Instead, it captures and stores the communication between the user and the pos daemon as a plain sequence of instructions. While this format remains readable, it may be harder to interpret, as it lacks the original structuring logic, such as loops and conditions, which are unpacked into a linear sequence of commands. However, even if a researcher does not take the extra step of documenting dependencies for the malleable reproduction approach, the experiment remains re-executable, and its artifacts can still be

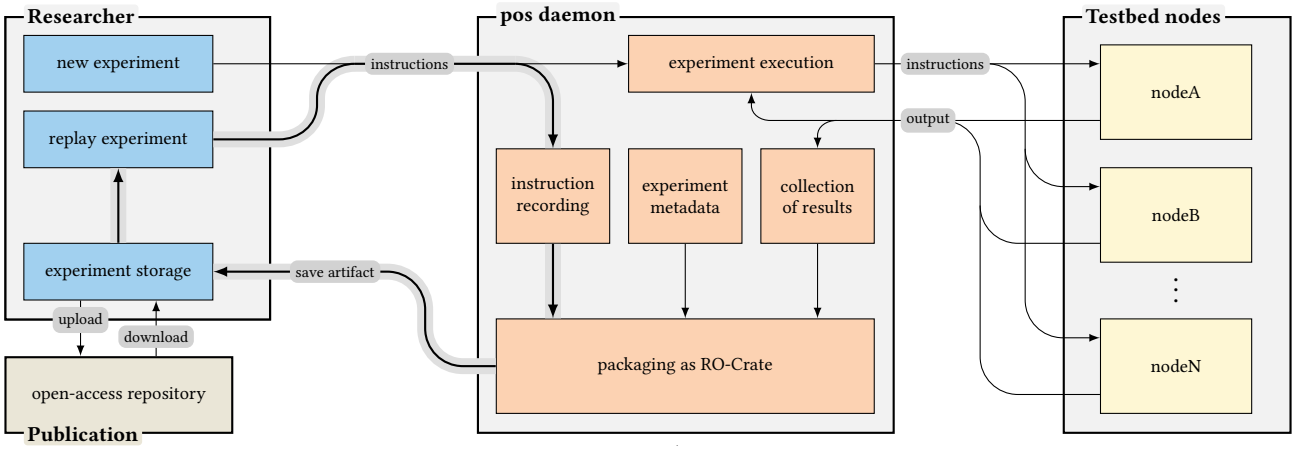


Figure 3: Workflow of the record-replay approach: The recording occurs independently of the researcher and takes place in the pos daemon. The closed-loop process is highlighted.

accurately reproduced using this method. This ensures that even in cases where researchers do not plan for reproducibility in advance, their experiments can still be replayed, preserving their results and making them available for future validation.

Figure 3 highlights the integration of the record-replay approach in the overall pos workflow. Analogously to the malleable reproduction approach in Section 5, this approach also introduces a closed-loop process. The process begins in the top left corner of the figure with the creation of a new experiment. As the experiment execution progresses at the top of the figure, all requests sent to the pos daemon are recorded. While the pos daemon executes the instructions, the recorded instructions are continuously dumped in the respective experiment RO-Crate, as shown in the center of the figure. Once the experiment concludes, the complete RO-Crate is stored locally on the testbed's management node. Users can then publish and upload the results to an open repository, as depicted in the bottom left corner. Similarly, the RO-Crate can be downloaded for local storage and future use.

When a user initiates the replay of an experiment, the record-replay loop is completed. The thick arrows in the figure highlight this closed loop. The following sections provide a detailed explanation of the record and replay process.

6.1 Recording of Experiments

In general, researchers use a REST API for communication with the pos daemon. Through this API, researchers can send requests to the pos daemon, which include general instructions for tasks such as resetting testbed nodes and launching scripts on them. Hence, each experiment consists of a set of instructions directing the daemon which actions to perform with the testbed nodes. Currently, while the issued instructions are executed, they are not recorded.

Therefore, we extend the pos daemon to record all issued instructions, including their exact payloads. By implementing this feature on the daemon side, the recording runs in the background and, therefore, accomplishes requirement R-R1. The content of the payloads, as well as the structure, varies depending on the type of instruction. For example, a node reset request only needs the node's name, while a script execution request includes the full script in the payload. The same applies to file transfers to testbed nodes. Moreover, we include a timestamp for every request. These timestamps can be used to estimate the experiment's execution time. Finally, the complete sequence of instructions is then stored in a JSON file located in the experiment's RO-Crate instance.

6.2 Replaying of Experiments

After an experiment has been completed, its replay can be initiated. Therefore, the researcher uses a new CLI command of MARTE on the testbed's management node, which reads the recorded sequence of instructions from the recording's JSON file stored in the experiment's RO-Crate. Next, this CLI command (i.e. replay tool) transmits the instructions together with their payload to the pos daemon. The daemon processes the received instructions identically compared to a regular researcher's input, reproducing the experiment under the same conditions as in the original execution. By maintaining the exact execution order, the replay mechanism ensures that the experiment runs identically, eliminating any inconsistencies between original and replayed execution. As the replay tool manages the entire process, we meet the requirement R-R2 for one-click replay.

In conclusion, the replay approach enables researchers to validate their results by re-executing their experiments. Since every executed instruction is stored in the RO-Crate, the replay process requires no additional manual input, making it a seamless and fully automated part of MARTE and the pos framework.

7 Publication and Sharing of Experiments

In the current implementation of pos, all experiment results are stored solely in a predefined location on the pos management host's local storage. Moreover, after an experiment has concluded, no further management of the results is available. Therefore, the further management of results is handled individually by every researcher. As a consequence, result management is a tedious process requiring manual action from the researcher. This manual handling also leads to many different approaches rather than a standardized process.

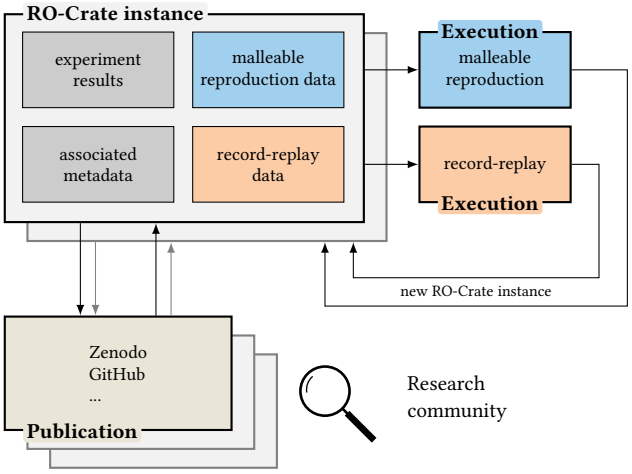


Figure 4: Publication, sharing, and re-execution of artifacts with MARTE.

To address this gap in result management, we propose an automated process for publishing experimental artifacts. We aim to create a framework that enables the community to share results and build upon existing work, thereby promoting collaboration. This approach is analogous to GitHub's philosophy, in which users can share their code publicly, create forks to enhance existing work, and then can contribute back to the community. As pos integrates with other RIs [24], researchers can choose their preferred environment and test across infrastructures, satisfying requirement R-G3.

For the implementation of our approach, we retain the local storage while introducing additional tools to improve result management. Specifically, we will enable the automated publishing of experimental results to external open repositories such as Zenodo and GitHub to facilitate seamless artifact uploads. Therefore, we extend the pos methodology with a result management endpoint to easily upload results in a standardized process. As part of this process, researchers can specify the title of the publication, add additional authors etc., fulfilling the requirement outlined in R-G2.

In general, the connection to online repositories offers several key advantages: (1) Results are automatically backed up to an external, persistent storage location, reducing the risk of data loss. (2) Experiments published on Zenodo receive a DOI, making them uniquely identifiable and citable in academic research. (3) Sharing results in publicly accessible repositories fosters open science and FAIR, allowing researchers to discover, analyze, reproduce, or modify existing experiments.

Figure 4 illustrates this process from the perspective of a researcher. A researcher can either search through online platforms for available experiment artifacts or authors of scientific publications provide the DOI to their experiment artifact. Platforms like GitHub or Zenodo allow users to directly view the artifacts online, and due to the structured and linked approach of RO-Crate, the individual files are well documented. If researchers wish to explore the artifacts further, they can download them to their local RI storage.

In the MARTE implementation, the downloaded artifacts consist of either three or four components. First, there are the results and the associated metadata of the experiment. Next, the contents of the record-replay approach, which were automatically created during the initial execution of the experiment, are included. If the author has created the configuration file for the malleable reproduction approach, this information is also available in the artifact.

Researchers focused on validating results can use the record-replay method. In contrast, researchers who want to understand the structure and execution of the experiment or modify it should prefer the malleable reproduction method.

In both cases, re-executing the experiment generates a new RO-Crate instance as depicted in Figure 4. This way, both the original RO-Crate and the new instance provide all the information necessary for a precise comparison between the original and the re-created results. The new RO-Crate instance can then be uploaded and published as well, completing the loop between experiment creation and re-execution.

8 Demonstration

We present a demonstration experiment to showcase the MARTE implementation. This demonstration consists of multiple parts.
First, we create a new experiment in the SLICES/pos RI. We decide to perform a network measurement to create an experiment that involves multiple nodes and is sufficiently rich to meaningfully test MARTE's reproducibility capabilities. In this measurement, we evaluate the performance of different implementations of the transport protocol QUIC [16]. After the measurement, we briefly present the results and publish the result artifact on Zenodo.
Next, we demonstrate the functionality of the two MARTE approaches: malleable reproduction and automated record-replay. We begin by using the automated record-replay to validate the results on the same servers, relying solely on the recorded information contained in the published artifact. Thereafter, we use the malleable reproduction, enabling us to modify the original measurement scripts as part of the artifact. To demonstrate the experiment's malleability, we switch to a different server pair, which requires a minor modification of the original experiment due to changed network interface names. Finally, we compare the original measurement with the replayed and reproduced measurements.

8.1 Experiment Description

QUIC is a transport protocol standardized by the Internet Engineering Task Force (IETF) in 2021 [16]. It builds on top of the User Datagram Protocol (UDP) and is designed to provide reliable, secure, and low-latency transport. By integrating Transport Layer Security (TLS) directly into the protocol, QUIC eliminates the need for a separate TLS handshake. QUIC is used as transport protocol in the HTTP/3 standard [2]. Unlike the Transmission Control Protocol (TCP), QUIC is usually implemented in user space. While this allows for faster development cycles, it results in numerous implementations in different programming languages. For our demonstration experiment, we use two popular open-source implementations (both written in C) of QUIC: *lsquic* [14] and *picoquic* [3]. As the purpose of our experiment is to demonstrate and test the MARTE framework, the selection of the QUIC implementations is arbitrary.

For the measurement, we use two identical off-the-shelf servers that are directly connected over an Ethernet Link. Specifically, the

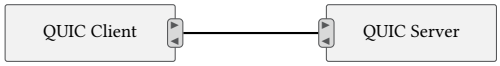


Figure 5: Demonstration experiment setup.

servers are equipped with a quad-core *Intel Xeon E3-1230* CPU, 16 GB RAM, and a 10 Gbit/s *Intel X540* network interface card (NIC). One machine acts as client, while the other acts as server. Figure 5 illustrates the described experiment setup.

We use the provided example implementations of both QUIC libraries to perform the measurements. The client requests a single file and then downloads it from the server via HTTP. After the transmission is completed, the client and server are stopped. We use the goodput, defined as the file size divided by the client's runtime, as the primary performance metric. This measurement procedure is simplified but comparable to the one presented before by Kempf et al. [18].

8.2 Experiment Execution

The experiment is implemented using the pos methodology and, therefore, split into three parts. The setup part includes the configuration of the nodes themselves, the network interfaces, and the installation of required packages. The main part, the experiment itself, starts with copying variables onto the nodes and building the QUIC implementations. The loop functionality of pos is used to execute the actual measurement multiple times with different parameters. In each loop iteration, from now on referred to as run, a new certificate and testfile are created on the server to ensure equal conditions for each run. We added a few loop variables to the experiment, including the file size of the transferred file, the used QUIC implementation, and a bandwidth limit, which is emulated on the server for outgoing traffic. Each configuration of loop variables can be repeated to achieve a higher confidence in the results by averaging the measured values. For our demo, a value of ten repetitions is used. During the transmission, logs are collected at both endpoints. The log directory is uploaded to the management node after each run to become part of the experiment artifact. The evaluation part of the experiment includes log analysis and transformation as well as plot generation. For this part, the collected logs, metrics, and used variable sets from all runs are copied onto one measurement node, where the analysis is performed. We implemented a basic evaluation that calculates mean average values as well as standard deviations from goodput values over all repetitions. Finally, a sample plot visualizing the measurement results is generated and uploaded to the management node, together with a summary of the results.

8.3 Experiment Evaluation

Network experiments, in particular involving complex protocols such as QUIC, provide both an ideal and challenging setting for evaluating reproducibility. In comparison to a simple calculation, yielding the same result every time, network experiments are much more complex and, therefore, more difficult to reproduce. To address this, we only look at the goodput and the general trend for different emulated bandwidths. While measured results vary slightly even with identical parameters, the outcomes of the same experiment

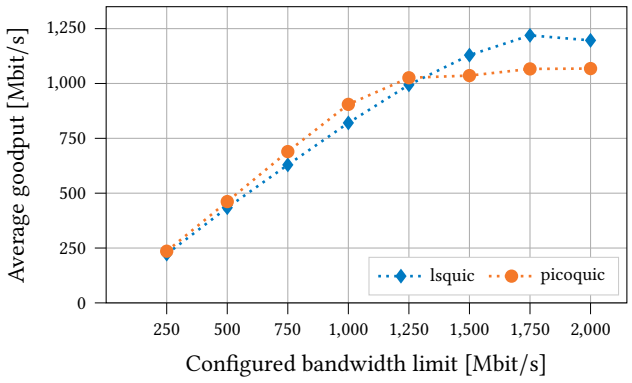


Figure 6: Original measurement showing the performance of two QUIC implementations.

on different nodes are comparable and lead to the same conclusion, as previously shown [18].

The results from the original measurement are shown in Figure 6. The x-axis depicts the configured bandwidth limit on the server. The y-axis displays the measured average goodput over ten repetitions for the implementations. For the lower bandwidth limits until 1250 Mbit/s, we observe that *picoquic* achieves a minimal higher goodput than *lsquic*. With a higher bandwidth limit, *lsquic* reaches a significantly higher goodput of 1196 Mbit/s compared to 1068 Mbit/s of *picoquic*.

8.4 Publication

The complete artifact of the demonstration experiment is available at Zenodo: https://doi.org/10.5281/zenodo.15747497. As described in Section 7, the publication is a seamless and automated process. It can be initiated anytime after the experiment has terminated. All contents of the published artifact are automatically generated during the execution of the experiment, with no manual additions. The artifact contains the experiment results, metadata, malleable reproduction scripts, and the automated recording. Formatted as RO-Crate, the artifact's contents align with Figure 4.

Artifact Explanation. To simplify the exploration of the artifact, we want to highlight the location of the aforementioned contents in this paragraph. On the root of the artifact, we have the mandatory `ro-crate-metadata.json`, which describes and links all contents of the artifact. The scripts and their outputs, which have been executed on the experiment nodes, are stored in the `scripts` folder. The result files, including the goodput plot of the experiment, are located in the `files` folder. All information about the environment, such as hardware and topology, are stored in `config` folder.

The `malleable_reproduction` folder contains the original experiment scripts and the configuration file (`config.yml`) of the malleable reproduction approach. The experiment recording is stored in the `experiment_recording.json` file.

8.5 Experiment Reproduction

In this section, we want to showcase the automated reproduction of the published experiment artifact powered by MARTE.

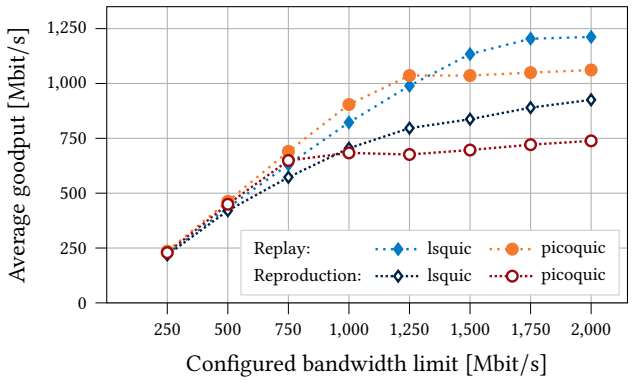


Figure 7: Reproduced measurements with MARTE showing the performance of two QUIC implementations.

First, we use the automated record-replay method to seamlessly replay the original experiment. As the record-replay approach is unsuitable for modifying the recorded experiment, we replay it on the same testbed nodes. Therefore, the objective of the replay is to accurately validate the result of the original measurement.
Second, to demonstrate the malleability of the experiment, we modify the original measurement files. Therefore, we switch to a different pair of servers. For these servers, the interface names have changed; thus, we need to change a variable file of the experiment scripts. The first server, acting as the QUIC server, is equipped with an *Intel Xeon E5-2640 v2* CPU and 32 GB of RAM. The second server, which functions as the QUIC client, is a dual-socket system using the same CPU model but has 16 GB of RAM. Both servers are equipped with NICs from the *Intel X500* family.

Reproduction Evaluation. Figure 7 depicts the results of the two experiment reproductions. We first look at the replayed results highlighted with the filled markers in the figure. In general, similar to the original measurement, for lower bandwidth limits the goodput of *picoquic* is minimally better than *lsquic* while for larger bandwidths *lsquic* is noticeably faster than *picoquic*. Moreover, if we look closely, all goodputs are very close to the original measurement, validating the initial results. If we pick one value, for example *lsquic* at the 1500 Mbit/s limit, the goodputs are 1129.15 Mbit/s for the original and 1133.96 Mbit/s for the replay, respectively. This results in a deviation of approx. 0.43 % between the original and replayed experiment.

In comparison, the malleable reproduction results are highlighted with the white-filled markers in the figure. We note that the goodput is worse than the original and replayed measurement. This effect can be explained by the different servers which were used for the reproduction. Since MARTE documents detailed hardware information in the artifacts, we can use this information to compare the different setups. Consequently, we observe that the CPUs of the original experiment have 4 cores with 3.2 GHz base clock, while the servers of the malleable reproduction have 8 cores but only a base clock of 2.0 GHz. Due to the reduced base clock and comparable architecture, the single core performance of the malleable reproduction servers is worse. As the used QUIC implementations are single-core applications, the lower goodput can be explained by the reduced single-core performance.

9 Conclusion

Reproducing and validating experimental results in the computation and data networking area remains a complex challenge due to intricate dependencies, evolving environments, and insufficient documentation. While research infrastructures provide a solid foundation for reproducibility, additional concepts and tools are required to boost their full potential.

We presented MARTE, a novel methodology to close the loop between experiment execution, result sharing, and reproduction. We made several contributions to implement this methodology. First, we package experiment results alongside with the scripts that generated them, obtaining self-contained experiment artifacts. We use RO-Crate as foundation for result data management to integrate structured and rich metadata. Next, we proposed two approaches for a user-friendly reproduction of experiments: malleable reproduction, which enables flexible reuse and adaptation of existing experiments, and record-replay, which captures and replays experiment executions fully automated for simple reproduction. By emphasizing malleability, experiments become shareable artifacts, allowing the research community to build upon existing work. Moreover, we provide tools to streamline the publication of artifacts to open repositories. For all the mentioned contributions, a key aspect of this work is automating the process as much as possible to minimize the effort required for reproducible research.

Future research could explore whether adding components in addition to cross-research infrastructure (cross-RI) compatibility would further improve reproducibility. We also envision creating a digital twin of the testbed or the identical experiment hardware setup to facilitate the reproduction of experiments. To implement this approach, MARTE is well-equipped due to its comprehensive environment documentation. Additionally, the machine-readable RO-Crate format simplifies the process of automated replication.

Our demonstration experiment shows that MARTE streamlines the reproduction process while decreasing manual effort. We showed that the measurement can be seamlessly validated using record-replay, and we applied the malleable reproduction approach to execute the experiment on a different hardware setup. We made the artifact of our demonstration experiment publicly available on Zenodo (https://doi.org/10.5281/zenodo.15747497) to allow the research community to better understand the advantages of the MARTE methodology.

Acknowledgments

We received funding from the EU's Horizon 2020 programme as part of the projects SLICES-PP (10107977) and GreenDIGIT (101131207). Moreover, we received funding from the Federal Ministry of Research, Technology and Space (BMFTR) under the projects 6G-life (16KISK002) and 6G-ANNA (16KISK107) as well as from the German Research Foundation (DFG) within the project HyperNIC (CA595/13-1). Additional funding was received from the Bavarian Ministry of Economic Affairs, Regional Development and Energy within the project 6G Future Lab Bavaria.

References

[1] ACM. 2020. Artifact Review and Badging Version 1.1. https://www.acm.org/publications/policies/artifact-review-and-badging-current Accessed: 2025-03-25.

[2] Mike Bishop. 2022. HTTP/3. RFC 9114. doi:10.17487/RFC9114

[3] Christian Huitema. 2017. picoquic. https://github.com/private-octopus/picoquic Accessed: 2025-04-03.

[4] Chamelon Cloud. 2021. Trovi. https://trovi.chameleoncloud.org/dashboard/about Accessed: 2025-03-30.

[5] Christian Collberg, Todd Proebsting, Gina Moraila, Akash Shankaran, Zuoming Shi, and Alex M Warren. 2014. Measuring reproducibility in computer systems research. *Department of Computer Science, University of Arizona, Tech. Rep* 37 (2014).

[6] Dmitry Duplyakin, Robert Ricci, Aleksander Maricq, et al. 2019. The Design and Operation of CloudLab. In *2019 USENIX Annual Technical Conference, USENIX ATC 2019, Renton, WA, USA, July 10-12, 2019*. https://www.usenix.org/conference/atc19/presentation/duplyakin

[7] European Organization For Nuclear Research and OpenAIRE. 2013. Zenodo. doi:10.25495/7GXK-RD71

[8] Serge Fdida, Nikos Makris, Thanasis Korakis, Raffaele Bruno, Andrea Passarella, Panayiotis Andreou, Bartosz Belter, Cedric Crettaz, Walid Dabbous, Yuri Demchenko, and Raymond Knopp. 2022. SLICES, a scientific instrument for the networking community. *Comput. Commun.* 193 (2022), 189–203. doi:10.1016/J.COMCOM.2022.07.019

[9] Sebastian Gallenmüller, Dominik Scholz, Henning Stubbe, and Georg Carle. 2021. 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. doi:10.1145/3485983.3494841

[10] Yolanda Gil, Ewa Deelman, Mark H. Ellisman, Thomas Fahringer, Geoffrey C. Fox, Dennis Gannon, Carole A. Goble, Miron Livny, Luc Moreau, and Jim Myers. 2007. Examining the Challenges of Scientific Workflows. *Computer* 40, 12 (2007), 24–32. doi:10.1109/MC.2007.421

[11] Eric Hauser, Sebastian Gallenmüller, and Georg Carle. 2024. RO-Crate for Testbeds: Automated Packaging of Experimental Results. In *IFIP Networking Conference, IFIP Networking 2024, Thessaloniki, Greece, June 3-6, 2024*. IEEE, 654–659. doi:10.23919/IFIPNETWORKING62109.2024.10619057

[12] Fabien Hermenier and Robert Ricci. 2012. How to Build a Better Testbed: Lessons from a Decade of Network Experiments on Emulab. In *Testbeds and Research Infrastructure. Development of Networks and Communities - 8th International ICST Conference, TridentCom 2012, Thessanoliki, Greece, June 11-13, 2012, Revised Selected Papers (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Vol. 44)*, Thanasis Korakis, Michael Zink, and Maximilian Ott (Eds.). Springer, 287–304. doi:10.1007/978-3-642-35576-9_24

[13] Tobias Hummel and Johannes Manner. 2024. A Literature Review on Reproducibility Studies in Computer Science (short paper). In *Proceedings of the 16th ZEUS Workshop, Ulm, Germany, February 29-March 1, 2024 (CEUR Workshop Proceedings, Vol. 3673)*, Sebastian Böhm and Daniel Lübke (Eds.). CEUR-WS.org, 54–62. https://ceur-ws.org/Vol-3673/paper9.pdf

[14] LiteSpeed Technologies Inc. 2017. lsquic. https://github.com/litespeedtech/lsquic Accessed: 2025-04-03.

[15] ORCID Inc. 2010. ORCID: Open Researcher and Contributor ID. https://orcid.org/. Accessed: 2025-06-21.

[16] Jana Iyengar and Martin Thomson. 2021. QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000. doi:10.17487/RFC9000

[17] Kate Keahey, Jason Anderson, Zhuo Zhen, et al. 2020. Lessons Learned from the Chameleon Testbed. In *2020 USENIX Annual Technical Conference, USENIX ATC 2020, July 15-17, 2020*. USENIX Association. https://www.usenix.org/conference/atc20/presentation/keahey

[18] Marcel Kempf, Benedikt Jaeger, Johannes Zirngibl, Kevin Ploch, and Georg Carle. 2024. QUIC on the Fast Lane: Extending Performance Evaluations on High-rate Links. *Comput. Commun.* 223 (2024), 90–100. doi:10.1016/J.COMCOM.2024.04.038

[19] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian E. Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica B. Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, and Jupyter Development Team. 2016. Jupyter Notebooks - a publishing format for reproducible computational workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas, 20th International Conference on Electronic Publishing, Göttingen, Germany, June 7-9, 2016*, Fernando Loizides and Birgit Schmidt (Eds.). IOS Press, 87–90. doi:10.3233/978-1-61499-649-1-87

[20] California Digital Library, Crossref, and DataCite. 2019. ROR: Research Organization Registry. https://ror.org/. Accessed: 2025-06-21.

[21] Rudolf Mayer and Andreas Rauber. 2015. A Quantitative Study on the Re-executability of Publicly Shared Scientific Workflows. In *11th IEEE International Conference on e-Science, e-Science 2015, Munich, Germany, August 31 - September 4, 2015*. IEEE Computer Society, 312–321. doi:10.1109/ESCIENCE.2015.58

[22] Daniel Rosendo, Kate Keahey, Alexandru Costan, Matthieu Simonin, Patrick Valduriez, and Gabriel Antoniu. 2023. KheOps: Cost-effective Repeatability, Reproducibility, and Replicability of Edge-to-Cloud Experiments. In *Proceedings of the 2023 ACM Conference on Reproducibility and Replicability, ACM-REP 2023, Santa Cruz, CA, USA, June 27-29, 2023*. ACM, 62–73. doi:10.1145/3589806.3600032

[23] Stian Soiland-Reyes, Peter Sefton, Mercè Crosas, et al. 2022. Packaging research artefacts with RO-Crate. *Data Sci.* 5, 2 (2022). doi:10.3233/DS-210053

[24] Henning Stubbe, Sebastian Gallenmüller, and Georg Carle. 2024. The Pos Experiment Controller: Reproducible & Portable Network Experiments. In *19th Wireless On-Demand Network Systems and Services Conference, WONS 2024, Chamonix, France, January 29-31, 2024*. IEEE, 85–92. doi:10.23919/WONS60642.2024.10449532

[25] W3C. 2020. JSON-LD 1.1: A JSON-based Serialization for Linked Data. https://www.w3.org/TR/json-ld/ Accessed: 2025-03-25.

[26] Mark D Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E Bourne, et al. 2016. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific data* 3, 1 (2016).