IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 25 February 2026, accepted 18 March 2026, date of publication 6 April 2026, date of current version 10 April 2026.

Digital Object Identifier 10.1109/ACCESS.2026.3680994

RESEARCH ARTICLE

Reproducibility of Experiments in Networking

NIKOLAS WEHNER[1], ALEXEJ GRIGORJEW[1], SIMON RAFFECK[1], MICHAEL GOEDICKE[2], AND TOBIAS HOßFELD[1], (Senior Member, IEEE)

[1]University of Würzburg, 97070 Würzburg, Germany
[2]University of Duisburg–Essen, 45141 Essen, Germany

Corresponding author: Nikolas Wehner (nikolas.wehner@uni-wuerzburg.de)

This work was supported in part by the Deutsche Forschungsgemeinschaft (DFG) in the Project NFDIxCS under Project 501930651.

ABSTRACT Reproducibility is a cornerstone of good scientific practice, enabling validation of experimental results and fostering progress by building upon prior work. The importance of reproducibility in networking research has gained traction in recent years. Notably, the Dagstuhl beginner's guide to reproducible experimental networking research has provided valuable best practices for documentation, data handling, and experiment design. However, the identification of specific networking components and aspects that prevent experiment reproducibility has not been investigated yet. This work addresses this gap by systematically identifying such components, constructing a taxonomy based on the findings, and proposing guidelines to increase experiment reproducibility. For this purpose, we discuss and analyze several networking use cases: 5G measurement experiments, Internet measurements, AI/ML-based networking experiments, and subjective QoE studies. Through targeted experiments and a review of existing studies, we identify some key sources of variability and discuss strategies to reduce variability and thus increase reproducibility. Our analysis highlights the importance of clearly defining research questions and expected outcomes as they form the foundation for assessing reproducibility. Additionally, our findings show the need for documenting potential sources of randomness, e.g., platform-specific effects introduced by operating systems, prior to experimental design. Minimizing unnecessary variability, e.g., by abstracting or replacing such components, further enhances reproducibility. While not all variability can be controlled due to the inherent complexity of networked systems, the proposed guidelines offer practical steps toward more reproducible networking experiments.

INDEX TERMS Reproducibility, Networking, 5G, Internet, AI/ML, QoE.

I. INTRODUCTION

Reproducibility of experiments is critical for good scientific practice. It not only allows to validate the experiments of other researchers, but also to build on top of existing works. Reproducibility is therefore an important topic in all fields of science. Formally, reproducibility is defined as a different research team repeating experiments on a different hardware setup and obtaining the same results with a stated precision [1]. In the context of networking, the consideration and importance of reproducibility also gained traction in the last years. Reproducibility for networking experiments was therefore covered in a Dagstuhl seminar, after which the Dagstuhl beginner's guide to reproducible experimental networking research had been published [1]. In this beginner's guide Bajpai et al. mainly cover general best practices in terms of problem formulation, documentation, data collection, and data handling. Additionally, they discuss how to document different experiments, e.g., simulations, systems prototyping, real-world measurements, and human experiments, and which software tools are recommended in order to increase reproducibility. While their work is highly valuable for researchers, we feel like one critical aspect is missing, namely, the identification of the technical components and aspects of the networking stack that actually prevent experiment reproducibility. Prior studies have generally focused on specific tasks, e.g., TSN [2], or provided high-level guidelines, such as the Dagstuhl report [1], without discussing the underlying technical mechanisms. In contrast, our work systematically identifies these technical

The associate editor coordinating the review of this manuscript and approving it for publication was Miguel López-Benítez.

© 2026 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

components and aspects, generates a taxonomy from these findings, and provides practical guidelines to mitigate these limitations, with the goal of enabling reproducibility across diverse experiments. By focusing on the technical sources of variability, our approach offers a more detailed and actionable framework than prior frameworks.

For our analysis, we cover four common use cases of networking experiments in line with the Dagstuhl beginner's guide, namely, 5G measurement experiments (system prototyping), Internet/local measurements (real-world measurements), AI/ML-based networking experiments (incorporating random numbers and stochastic processes similar to simulations), and Quality of Experience (QoE) experiments (including human feedback). For each use case, we conduct simple experiments if possible or reuse existing works, and identify the components or aspects responsible for result variability. Afterwards, we discuss how these components have to be adjusted to achieve reproducibility with respect to the obtained results.

Our key contributions can be summarized as follows: We show that physical signals, in particular in the wireless case, must be carefully handled to obtain controllable networking experiments whenever they are involved, e.g., by switching to direct links, emulating signals, or strictly eliminating interferences. Otherwise, these uncontrolled variances may impact result reproducibility. When conducting Internet measurements, we experience the highest degree of variability as several components and aspects are not controllable. To obtain reproducible experiments, the testbed has to be adjusted to become closed and controllable, where custom services must replace the actual service. Considering AI/ML-based experiments, we note that these experiments are reproducible as long as everything is software-based, i.e., no new measurements have to be performed, and fairly independent of random seeds. Finally, we remark that even though the identified aspects are controlled for the respective experiments, the variability of the Device Under Test (DUT) may still limit or prevent reproducibility, e.g., due to background apps or unusually high CPU load, and therefore has to be strictly monitored during experiment execution. Summarizing, we provide a taxonomy and guideline for future researchers, which can be used to design reproducible experiments and control variances introduced by networking components.

This work is structured as follows: In Section II, we first discuss related work. Afterwards, we discuss the use cases in a bottom-up fashion, starting with 5G networking experiments on physical layer (Section III), followed by Internet measurements (Section IV), before we conclude with AI/ML-based networking experiments (Section V). We summarize our findings in Section VII, where we also establish a taxonomy as guideline for future networking experiments, before we conclude in Section VIII.

II. RELATED WORK

A multitude of definitions for reproducibility exist in research, e.g., by ACM [3], [4] or by the National Academies of Science [5]. In this work, we reuse the definition of reproducibility according to the Dagstuhl beginner's guide [1], in which repeatability, replicability, and reproducibility are defined as follows:

Repeatability: The experiment can be reliably repeated by the same researcher to produce the same results (same team, same experimental setup).

Replicability: An experiment can be reliably repeated by a researcher reusing the experiment setup of another researcher to produce the same results (different team, same experimental setup).

Reproducibility: The same results can be obtained for a different experiment setup and under different conditions by an independent researcher (different team, different experimental setup).

Note that the definitions of replicability and reproducibility have been exchanged in 2020 in [3] and in [4], leading to general confusion about the two definitions. In this work, we adopt the old definition [4], stated above and also used by the Dagstuhl's beginner guide [1]. Further, the definitions lack clarity on what same and different experimental setup actually means and at which point a technical setup can be considered the same, e.g., does a slightly different CPU qualify as a different setup even though all remaining components are identical? These ambiguities remain open questions in literature.

In this work, we further focus on the perspective of reproducing results rather than methodologies. This means that necessary tools are provided for the experiment, eliminating the need to rebuild tools from scratch based on scientific papers, and only experiment results are expected to be reproduced.

Different works have addressed the quantification of the degree of reproducibility. Some works assign reproducibility scores according to specific, pre-defined criteria, e.g., how well are method, data, and experiment documented in an AI paper [6] or how well are dataset, experiment design, and evaluation aspects of a web measurement study documented [7]. Similarly, Cimiano et al. [8] define five levels of analytical reproducibility:

1) No data or software is available.
2) Data and code are available, but results can not be reproduced based on these artifacts.
3) Data and code are available and results are generally reproducible, but reproducibility is limited due to the requirement of commercial or proprietary software, uncommon data formats, or the requirement of guidance by original researchers.
4) The results are fully reproducible without any additional requirements.
5) The results are fully reproducible over a long period of time, i.e., sustainable reproducibility.

Several frameworks aim to ease networking experimentation and improve reproducibility. WaIT [9] targets wireless networks and leverages low-cost hardware and

Docker to enable customizable and shareable experiments across diverse scenarios. In contrast, the plain orchestrating service (pos) [10] is mostly used for wired networks and provides a fully scripted workflow and comprehensive documentation to ease recreation of experiments. Similarly, EnGINE [2], [11] supports wired, time-sensitive networking (TSN) experiments, using commercial off-the-shelf hardware and Ansible-based orchestration to evaluate TSN topologies and configurations in the context of vehicular networking.

Reproducibility has also been studied extensively in web measurements. Demir et al. [7], [12] show that even minor differences in experimental setups can prevent reproducibility and that identical setups may still yield non-repeatable results, often due to third-party web content beyond researchers' control. In a similar line of research, Hantke et al. [13] demonstrate that web archive-based measurements can improve reproducibility compared to live measurements, but with certain limitations.

III. USE CASE: 5G EXPERIMENTS

In the first use case, we consider two distinct, but similar 5G campus networks, located at the Norwegian University of Science and Technology (NTNU) and University of Würzburg (UWUE), with comparable but not identical hardware as described in Table 1 [14]. We thus focus on reproducibility of network performance on physical layer between testbeds. For this purpose, we conduct measurements in both sites, in which we collect network KPIs like throughput and one-way delay for measurements between an application and different gNodeB (gNB) implementations (SRS, OAI). We analyze mean uplink and downlink throughput for the gNB implementations in Table 2 with different downlink to uplink ratios. For UWUE, we investigated two setups with different elevations between gNB and UE (1 for elevated gNB and 2 for identical elevations). The table highlights mean throughput and 95% confidence intervals, with green indicating overlapping intervals and red marking statistically significant differences. The findings reveal notable performance differences between gNB implementations across testbeds, despite identical configurations and dedicated frequency bands. Unaccounted variables, such as environmental factors or interference with other radio signals, likely contributed to these variations. While the small sample size of scenarios may seem to limit generalizability here, the reported confidence intervals already provide strong evidence of poor reproducibility. These findings are further supported by subsequent studies independently confirming them.

TABLE 1. Testbed hardware setup.

	NTNU	UWUE
gNB/UE Host	Intel Core i7-6700	AMD Ryzen 7 3700X
SDR	USRP B210; UHD Version 4.0	
gNB Version	OAI: v2.1.0 (9fab212441) SRS: 24.04.0 (c33cacba7)	
Modem	Quectel RM520N-GL	
Frequency	3600-3640 MHz	

An observed reason for variability in the setup was the physical environment in which testing occurs. For example, the distance between UE and gNB seems to have led to signal attenuation, resulting in variations in throughput, latency, or other measurements. Further, we have seen in the UWUE setup that moving the gNB onto the same level as the UE results in a significant uplink speed increase by about 600% (scenarios 1 and 2). We further assume that walls, furniture, or other physical barriers additionally degraded the signal quality. All these effects thus need to be taken into account during measurements, as they are inherently variable across different testbeds and not entirely avoidable.

The actual load of the DUT (UE/gNB) may also affect performance. For instance, if the UE is handling additional applications or background processes during testing, this could influence its responsiveness or network throughput. Similarly, fluctuations in the load on the RAN, such as concurrent connections or system resource utilization, can introduce variability in test results, making it challenging to isolate and measure specific performance parameters.

Another reason for differences may be the fact that present cross-traffic is not comparable between the two testbeds. Such cross-traffic or interferences may include signals from neighboring wireless networks, devices operating on the same frequency bands, or even electromagnetic interference from non-communication equipment. Cross-traffic is often uncontrollable and can thus affect the study outcome, leading to inconsistencies in the measured performance. This factor emphasizes the importance of minimizing cross-traffic or testing in a controlled environment.

As previously discussed, the positioning of the gNB relative to the UE can significantly affect performance. Therefore, it is important to find the optimal settings of all relevant testbed parameters. Each testbed has unique characteristics, such as hardware configurations, antenna placements, and propagation conditions. Thus, achieving optimal performance often requires testbed-specific calibration. Experts in the field usually solve this problem with a binary search to fine-tune the dBm settings for the best performance. Identical configurations may thus not yield comparable results in different setups, providing another source of variability and limiting reproducibility.

Finally, external environmental conditions, such as temperature, humidity, and the presence of people, can theoretically affect network performance. However, these factors are often difficult to control, especially in real-world testing environments, further contributing to variability.

IV. USE CASE: LOCAL/INTERNET MEASUREMENTS

In this use case, we consider network measurements, which may involve the Internet and its services. As representative scenarios for such measurements, we consider simple network-only ping measurements, file download, and video service measurements, comprising both network and application layer metrics. From an abstract point of view, these use cases contain three stakeholders: a client, which

TABLE 2. Comparison of Mean Throughput and 95% confidence interval for 20 MHz in both testbeds for OAI and SRS. Two physical layouts for UWUE testbed [14].

gNB	Ratio	DL (Mbps)			UL (Mbps)		
		NTNU	UWUE (1), (2)		NTNU	UWUE (1), (2)	
SRS	1:1	42.81±1.03	42.33±1.34	42.87±1.04	8.96±3.90	3.30±0.70	18.05±2.23
	2:1	55.64±0.43	55.77±0.52	55.54±0.40	5.04±2.72	3.25±0.42	17.21±1.61
	4:1	59.86±3.50	60.62±3.85	60.02±3.52	5.94±3.14	2.11±0.42	13.98±0.10
OAI	1:1	42.20±0.84	39.12±1.97	38.37±1.83	11.31±4.42	11.66±4.16	9.51±3.37
	2:1	55.10±4.21	49.66±3.68	49.51±4.04	8.47±1.29	7.80±1.11	7.17±1.11
	4:1	39.25±1.13	36.42±1.24	35.87±0.80	8.25±0.07	7.90±0.08	7.13±0.10

initiates the pings, the file downloads, or the video playback, a server, which corresponds to the provider of the respective service, and the network, which corresponds to the end-to-end path between client and server, optionally routed through the Internet. We restrict the experiments to a local testbed only, i.e., without relying on the Internet. The implications of Internet measurements along with additional sources of variability are thus not covered in these experiments, but are discussed later in this section.

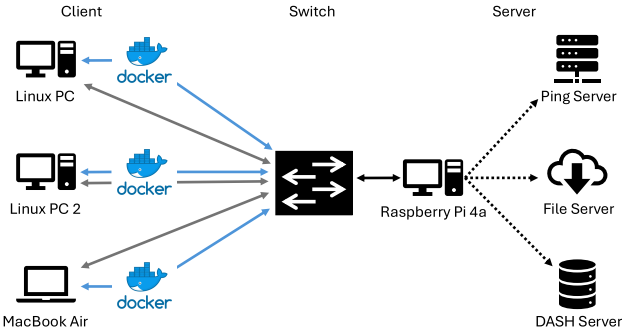


FIGURE 1. Local testbed for experiments.

A. EXPERIMENTS

Our experiment setup is depicted in Fig. 1 and consists of three client devices, namely, a Linux PC (AMD Ryzen 7 3700X, x86, Kubuntu 24.04), a second Linux PC (Intel Core i7-8700, x86, Ubuntu 24.04), and a MacBook (MacBook Air Apple M1 Chip, arm64, macOS 14.5 Sonoma), a switch, and a server, here a Raspberry Pi 4a. The client devices are connected via Ethernet to a single switch, which is again connected via Ethernet to our server, a Raspberry Pi 4a. The client devices are isolated during the experiments, with only one device active per experiment, ensuring the absence of cross-traffic. We run each experiment on the client device either in a non-virtualized environment (gray path) or within a virtualized setup using Docker (blue path). To hopefully ensure reproducibility in the virtualized environment, we use a consistent Dockerfile for the different devices. We use tcpdump to capture all traffic on the client's Ethernet interface. For the DASH experiments, we use Linux traffic control[1] on the server's outgoing Ethernet interface to emulate maximum bandwidth limits.

[1]https://man7.org/linux/man-pages/man8/tc.8.html

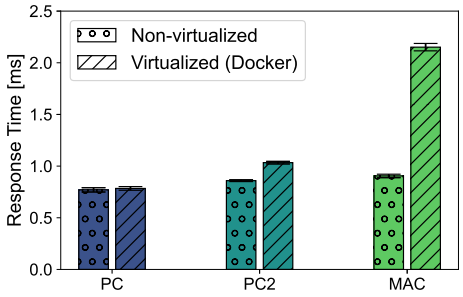


FIGURE 2. Results for ping measurements.

1) PING MEASUREMENTS

In this experiment, we perform simple ping measurements between client and server. We perform 10 measurement runs for each device (non-virtualized or virtualized), and each measurement consists of 30 subsequent pings. We depict the response times for the ping measurements for the non-virtualized environment and the dockerized environment in Fig. 2. The x-axis denotes the client devices, the y-axis the response time, and the bars depict the mean and the error bars the 95% confidence interval across the 300 pings (10 runs with 30 pings) for each scenario. The results show similar response times for the non-virtualized environment across the three devices, which comes as expected. If these ping measurements can be considered reproducible, however, depends on the formulated research question and its defined tolerance. If a research question investigates whether response times stay below 1 ms (high tolerance) across different setups, the measurements are reproducible. In contrast, if the research question investigates whether the obtained response times across measurement setups differ no more than 0.05 ms (low tolerance), the results from the measurements are not reproducible. Subsequently, it is critical to formulate precise research questions. In contrast to the non-virtualized setting, we observe significant differences between the MacBook (*MAC*) and the Linux devices in the virtualized setting, and also between the behavior of the MacBook in the non-virtualized and virtualized environment.

The primary difference lies in how Docker interacts with the host operating system [15]. On Linux, Docker runs natively, sharing the host kernel, whereas on macOS, it requires a Linux virtual machine (VM) to host the Docker engine and run containers, a capability provided

by Apple's Virtualization Framework. Networking between the host and container is also handled differently across operating systems. On Linux, the host kernel manages networking directly using virtual interfaces and namespaces, while on macOS a Docker-specific background service acts as a network proxy between the Linux VM and the host, forwarding traffic and performing NAT. These additional virtualization and proxying layers can introduce latency, likely contributing to observed performance differences here. Similar issues can be expected when comparing Linux and Windows, as Windows also requires a Linux VM provided via Hyper-V or WSL2.

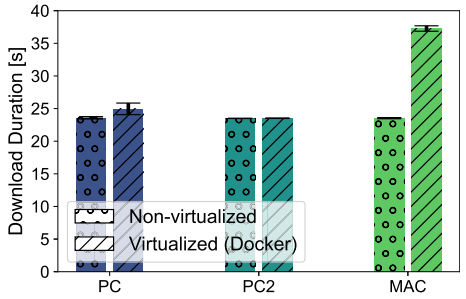


FIGURE 3. Download durations for file download.

2) FILE DOWNLOADS

This experiment consists of the client device downloading a video file of 240 MB from the HTTP file server with the Linux tool wget[2] and a maximum available bandwidth of 100 Mbps. In this experiment, we also perform 10 iterations for each device. Fig. 3 depicts the observed download durations for each client device in the non-virtualized and virtualized environment. Similar to the ping measurements, we observe reproducible results in the non-virtualized environment across devices, while the results differ between the devices in the virtualized environment. Again, these differences can be attributed to the negative impact of Docker, implying that the use of Docker images alone is not sufficient to create reproducible results. For the network layer, we show the observed mean packet sizes and inter-arrival times for uplink and downlink along with the 95% confidence intervals in form of error bars in Fig. 4. The figure shows that the highly similar devices *PC* and *PC2* also perform highly similar in terms of network behavior, yielding almost identical similar packet sizes and inter-arrival sizes for both uplink and downlink. In contrast, the device *MAC* used significantly lower downlink and uplink inter-arrival times and also lower packet sizes. Summarizing, while the download durations were reproducible across all devices in the non-virtualized environment, it does not guarantee the reproducibility of network-layer metrics. Since *PC* and *PC2* utilize a x64 CPU architecture with a Linux OS and the *MAC* utilizes a arm64 architecture with macOS, we attribute the observed differences on network layer to CPU architecture and operating system. Indeed, when analyzing the OS settings of TCP, which HTTP relies on, we can observe differences between *PC* and *MAC*, potentially leading to such discrepancies. As a consequence, it is necessary to accurately document the used CPU and OS. Further, the research question should be clearly defined to clarify if network behavior is of actual interest in terms of reproducibility or if application-layer metrics are sufficient for example for QoE experiments.

3) DASH

In this experiment, we prepare a DASH video along with a manifest on our HTTP server. We use the Big Buck Bunny video[3] with a framerate of 30 FPS and 9 different bitrates ranging from 200 kbit (video resolution of 180p) to 8000 kbit (video resolution of 1080p). On client side, the devices utilize the dash.js player[4] to stream the video from the server by requesting the corresponding DASH manifest. To log application layer metrics, e.g., stalling events, video quality, and video buffer, we integrate callbacks into the video player to periodically poll these metrics in one second intervals and write them to a file. In the DASH experiments, we limit the maximum available bandwidth per device to either 1 Mbps, 2 Mbps, or 5 Mbps. Again, we repeat all experiments, i.e., an experiment settings consisting the used of the used device and the set bandwidth limit, 10 times.

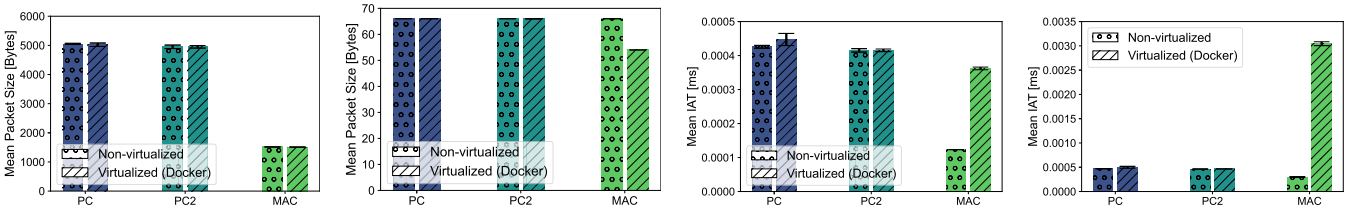
We report the application-layer metrics mean video resolution (in pixels) and the mean video buffer (in seconds) along with the network-layer metrics mean downlink and uplink packet inter-arrival times for the non-virtualized setting in Fig. 5. Here, the x-axis denotes the maximum bandwidth, the y-axis denotes the corresponding metrics, and the bars are colored according the used devices and represent the mean value along with the observed standard deviation across the 10 measurement runs in form of error bars. For the video resolution (cf. Fig. 5), we observe similar results across devices and maximum bandwidths, indicating reproducibility. However, if we investigate the video buffer behavior, we can again see differences between PCs and *MAC*, as the mean buffer of the *MAC* is generally lower than for the PCs, highlighting differences in the network behavior. In contrast to the file download scenario, the downlink packet inter-arrival times look rather identical for all devices this time, and the actual differences between PCs and *MAC* arise from the uplink packet inter-arrival times, where the PCs behave reproducibly, but the *MAC* behaves differently.

Note that we do not conduct the same experiment in the virtualized environment for the *MAC* as browser dependencies led to problems between x64 and arm64 devices. Google Chrome, used to access the video page, is only supported for x64 devices, thereby leading to problems of running the same Dockerfile. However, when utilizing the same Dockerfile

[2]https://www.gnu.org/software/wget

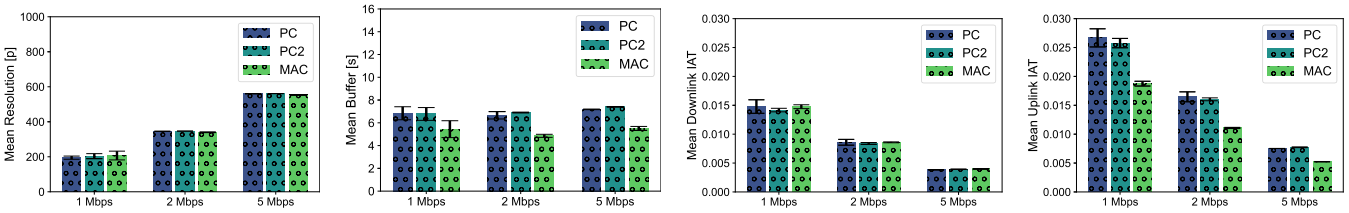
[3]https://dash.akamaized.net/akamai/bbb_30fps/

[4]https://dashjs.org/



(a) Downlink packet sizes. (b) Uplink packet sizes. (c) Downlink packet IAT. (d) Uplink packet IAT.

FIGURE 4. Network behavior for file download.



(a) Video resolution. (b) Video buffer. (c) Downlink packet IAT. (d) Uplink packet IAT.

FIGURE 5. Application and network behavior for DASH (non-virtualized).

between the PCs, we observed reproducible results. Nevertheless, our experiments show that it is important to keep CPU architecture and OS in mind when designing reproducible experiments as these factors can severely affect experiment outcomes, let alone the replicability of the methodology. Virtualization (with Docker) is not sufficient to circumvent such problems. As a final remark, it is again important to clarify the actual research question and which metric is supposed to be reproduced. If our goal had been to reproduce the played out video quality, our experiments would have been reproducible. If the goal had been the reproducibility of buffer behavior, however, our experiments would not have been reproducible across all devices.

B. OTHER REASONS FOR VARIABILITY

In the following, we discuss additional potential reasons for variability in measurements. While this list is likely not complete, it can be used as starting point to increase reproducibility in networking measurement experiments.

1) DEVICE VARIABILITY

Device variability refers to inconsistencies in the nodes responsible for monitoring or controlling network traffic. When using tools like tcpdump for packet capturing, packets may be dropped due to buffer overflow, particularly when the system is under heavy load. These losses may introduce inaccuracies in capturing the actual traffic patterns. Additionally, variability in the hardware or software configurations of these nodes, such as TCP settings, CPU load, or memory utilization, can affect performance and reliability of such tools.

Considering experiments, where control nodes manage interactions with devices under test (DUT), variability can arise from several sources. For instance, if the control PC is running multiple tasks simultaneously, delays in issuing commands like those via Android Debug Bridge (ADB) can occur for experiments where app performance is measured. Another problem may be the quality of the USB cable used to connect devices, where poor quality may introduce connectivity issues, leading to unreliable control over the DUT. Such factors can create unpredictable delays and affect the accuracy of measurements or test outcomes.

The DUT itself can also exhibit variability due to the impact of background applications running on the DUT. Such background applications may generate additional noise by consuming resources like CPU, memory, or network bandwidth. Subsequently, the performance and behavior of the targeted application, e.g., DASH, may be affected.

Similarly, virtualization can negatively affect experiment reproducibility, as CPU, memory, and I/O resources are often shared across process through namespaces and cgroups. Resulting resource contention and scheduling variability can lead to unpredictable performance, i.e., inconsistent network throughput and latency. Moreover, virtualization abstracts hardware details, potentially hiding critical system differences that affect results. Variations across operating systems and virtualization platforms further contribute to inconsistent behavior, ultimately reducing reproducibility.

2) WIRELESS COMMUNICATION AND INTERFERENCE

As we have already seen in the 5G experiments, signal interference may introduce additional experiment variability due to the shared nature of the medium. Signal interference from other devices operating in the same frequency band (e.g., Wi-Fi, Bluetooth, or IoT devices) can degrade performance. Sometimes, wireless protocols also adapt to network conditions, adding another layer of complexity to the

consistency of measurements. It is thus generally preferable to use wired connections if possible.

3) IMPLICATIONS OF INTERNET MEASUREMENTS

Internet measurements are subject to variability along the end-to-end path, as each device and network segment may introduce delays, jitter, or packet loss. Congestion at intermediate nodes, variations in routing paths, or differences in the performance of ISP networks can further affect the measurements. In distributed systems, these challenges become more complex as measurements must account for other traffic across multiple locations and differing network conditions.

4) CLOSED-BOX SERVICE BEHAVIOR

The behavior of online services being tested often depends on proprietary factors fully withheld from the researcher. For example, adaptive bitrate (ABR) algorithms in video streaming services can change dynamically based on perceived network conditions, making it challenging to predict and replicate service behavior. We want to remark that the verb *replicate* does not necessarily indicate *replicability* but only that the methodology is imitated. A/B testing, where services test different versions, introduces further variability as the results observed for one version may differ to the next version.

Another problem with closed-box services is the problem that content may change over time, leading to different data formats. For example, streaming platforms may update their encoding techniques, such as switching from H.264 to AV1, which can have significant impacts on network resource usage. These changes may occur without warning. Thus, when measuring closed-box services, the measurements always take a snapshot of the current system only, and are usually not comparable across time and thus often not reproducible.

Some apps, e.g., YouTube [16], may utilize historical information from previous sessions to reduce the network burden, but this may leave residual data or settings that impact subsequent tests. Such residual data may lead to inconsistencies in measurements. To reduce these effects, it is thus necessary to properly clean or isolate test setups to enhance reproducibility.

V. USE CASE: AI/ML-BASED EXPERIMENTS

In this use case, we consider the common AI/ML-based networking tasks anomaly detection and resource allocation as examples.

Usually, anomaly detection is treated as a classification task. The goal of a classification task is to infer the categorical target variable y, e.g., the type of intrusion attack, from a set of features X, e.g., network-layer features. In the supervised setting, the features X_i and the corresponding labels y_i are fed to the model so that it is able to learn the relationship between features and labels. In inference, the model then predicts the unseen labels $\hat{y}$ from the monitored features $\hat{X}$.

Resource allocation is a common reinforcement learning task, where the goal is to allocate the limited bandwidth resources to the end-devices in a network such that the network performance is optimized. The optimization target can be any arbitrary metric, e.g., channel utilization, number of supported end-devices, or users' QoE. With reinforcement learning, an agent performs an action in every step, e.g., increase or decrease the amount of resources for an end-device, and obtains a reward for the performed action. The agent then tries to learn a policy, which optimizes the total obtained reward. The policy directly decides which action the agent has to take in which state.

Even though supervised learning (SL) and reinforcement learning (RL) seem on the first glance rather unrelated, we show that both methodologies share all properties when it comes to reproducibility. In particular, both tasks are purely software-based, i.e., by default the methodology is easier to replicate and results are easier to reproduce than for hardware-based networking experiments.

For the AI/ML use case, we identify two main reasons for variability and thus for limited reproducibility, namely, data variability and inherent randomness of models, GPUs, and training procedure.

1) DATA VARIABILITY

Most data variability stems from the training data source. Independent of SL and RL, if live data is used for model training, reproducing those results with a new model is highly difficult to impossible. For the sake of reproducibility, training data, even when sourced from a live system, needs to be captured as a dataset and the dataset needs to be made available to allow reproducible model training. A proper ML methodology for both SL and RL requires a randomized split of the dataset into training, validation, and test set. While the training set is supposed to be used for model training only and consists of 70-80% of the data, the optional validation set (10-15% of the data) evaluates the model performance on different data over the course of the training. The test set (10-15% of the data) simulates an unknown dataset and its evaluation essentially quantifies model generalizability. Experiment results should thus be reported on the test dataset, but the model should be fitted towards training and validation set. A problem with reproducibility may now emerge when these dataset splits are not available to other researchers. Different test dataset splits may exhibit different properties, i.e., a different target distribution or different scales, and may thus lead to different results than in the original experiment. It is therefore fundamental to maintain the test dataset identical across experiments.

Different dataset splits also affect preprocessing steps and data transformations like standard scaling. Scalers are usually fitted on the training set only, and the fitted scalers are simply applied to the validation and test sets to transform the respective dataset. As the scaling depends on the train dataset only, wrong dataset splits might lead to a different scaling due to potential outliers and thus different results. Further,

specific neural network layers like Batch Normalization are generally tuned in training mode and then fixed in inference mode, i.e., these layers perform differently between training mode and inference mode. Training on different data than in the original experiment may therefore already lead to a different outcome.

To foster reproducibility in AI/ML-based networking experiments, the dataset splits thus always have to be made available along with the research paper, or at least the random seed used for splitting the data (for example in Python) has to be reported. Indeed, the Python interpreter shows consistent behavior across different hardware when a random seed is set. Currently, the unavailability of these dataset splits is a fundamental problem in advancing AI/ML-based networking experiments as it does not allow proper comparisons between different research works.

2) RANDOMNESS

Specific ML models possess inherent randomness. A good example for this are Random Forests, which randomly sample subsets of the training data, then train a decision tree for each of these subsets, and let the majority vote decide the outcome. Another example is the layer weight initialization of neural networks. For each layer containing weights, the weights are randomly initialized during model setup according to the selected layer weight initialization technique, e.g., random normal or random uniform. However, we want to emphasize that this randomness is inherent part of the models and should therefore lead to a similar outcome independent of the random seed. To counteract this randomness, which also prevents repeatability, it is therefore reasonable to conduct the experiment several times with different random seeds as the reproduced results should be on average similar compared to the original results. To obtain reproducible results though, setting the random seeds of the models and utilized toolkits is sufficient, e.g., SciKit-Learn provides a random state feature and Tensorflow/Keras also provides utilities to set random seeds [17].

Another source of randomness in AI/ML is hardware variability, particularly in GPU computations. GPUs can introduce small variations due to parallel execution and hardware-level optimizations [18], [19]. For example, the order of floating-point operations may differ between runs as threads complete asynchronously, and GPU libraries like cuDNN use optimized operations that are not always deterministic. While these variations are typically minor, they can affect reproducibility. However, frameworks like PyTorch and TensorFlow provide deterministic flags to enforce deterministic operations, mitigating the impact of GPU non-determinism.

Aligned with this discussion is the randomness introduced by the training data input pipeline. Models relying on gradient descent require randomness to converge. As a consequence, the data batches and the order of the data batches fed to the model during training are randomized, which however should not affect reproducibility, when considering the performance on average or training long enough. Nevertheless, setting random seeds in the toolkits should also provide repeatability and thus reproducibility.

While all previous aspects matched both SL and RL, RL adds the additional randomness of exploration and exploitation when the agent selects the next action, e.g., increasing resources for node A. In RL, agents randomly decide whether they follow the currently learned policy and select the best action (exploitation) or whether they randomly select one of the available actions (exploration). In line with the previous comments, obtained model performance should be on average similar to the model performance obtained in the original experiments, even if repeatability of the methodology is not given.

Summarizing, we note that supplying the used dataset splits in AI/ML-based networking experiments is fundamental to allow reproducibility as it may affect test set performance due to different samples or different scaling. Further, we observe that randomness in AI/ML mostly affects repeatability and that the stochastic nature of AI/ML algorithms and differences in numerical precision always lead to slightly different converged models. Nevertheless, when performing multiple training rounds, the obtained performance should be on average similar to the results obtained with the original model. To achieve repeatability and subsequently also reproducibility and replicability, existing models and toolkits also provide random state (seed) settings. Using these seeds, the Python interpreter causes consistent behavior across different hardware.

VI. USE CASE: SUBJECTIVE QOE STUDIES

In the following use case, we consider adaptive video QoE crowdsourcing studies, i.e., unknown users participate over a platform in a study, watch an adaptive video stream, and evaluate the subjectively perceived quality of the stream, i.e., Quality of Experience (QoE) [20]. These video streams are generally somehow modified such that they exhibit specific patterns in the video content, e.g., different video quality levels, quality switches, or stalling [21]. These emerging patterns then correspond to the stimuli. These modifications are usually either based on network traffic shaping or on custom video preparation. In the first case, network traffic is shaped in a way to force the desired stimuli, e.g., a bandwidth of 500 kbps to receive a video stream of low quality or force a stalling event. In contrast, in the second case, we would prepare the video source directly such that it is encoded in a low video quality and contains an emulated stalling event, which is then served to the participant. Participants then watch the served video unaware of any modifications and provide feedback, i.e., ratings, on the quality of the video streams. We emphasize that there is a technical and a human perspective, which may be cause for variability and thus affect reproducibility. The technical perspective includes the technical setup like end-devices, server, and the network, while the human perspective includes demographical and

psychological influence factors, e.g., age of participants, technical skills, mood or interest in study, time of day, or social factors.

First, we identify reasons for variability from the technical perspective, one for stimuli preparation and one for DUT variability. For the human perspective, we state that the human subject is simply another DUT, but that we cannot properly control it nor describe it due to unmeasurable factors (priors).

Preparing the study content beforehand, should be the preferred way of conducting crowdsourcing QoE studies to achieve reproducibility. We assume that network traffic shaping does not necessarily replicate the exact video stimuli for each participant. Subsequently, different stimuli between the original and the replicated study may result in different results and thus no result reproducibility, even though the methodology has been replicated.

Another reason for variability may be again the load on an end-device, e.g., caused by background apps or generally high CPU usage. This may affect the playback of the streamed video. In crowdsourcing studies, participant's end-devices are not controlled.

Summarizing, we observe that in crowdsourced QoE studies the modification and preparation of the actual stimuli plays a fundamental role in increasing reproducibility. We further observed that the use of different end-devices, e.g., caused by crowdsourcing, may also limit reproducibility as stimuli are differently perceived. Conversely, laboratory-based QoE studies offer significantly greater experimental control, thereby likely ensuring higher reproducibility, but typically involve fewer participants, limiting statistical significance and generalizability. This raises an important conceptual question: should reproducibility be prioritized over statistical significance, or vice versa? In other words, should reproducibility be understood as the ease of reproducing results under controlled experiment conditions, or as the consistency of findings across a broader range of (uncontrolled) variables. We argue that this reproducibility-generalizability trade-off can guide researchers in defining precise research questions that clarify which aspect of reproducibility is most important for a given study.

Finally, the human perspective often also plays a significant role in the study outcome. A key challenge is that the research team cannot control which specific aspects, such as demographic or psychological factors, participants will adhere to. Recruiting participants for laboratory studies is typically difficult, and finding a group for the reproducibility study that matches the characteristics of the participants in the original study is nearly impossible. An increase in the population size of the participants in both studies may resolve this issue by balancing out the different factors, e.g., using crowdsourcing. To mitigate such challenges in crowdsourcing studies, best practices should thus be followed [22], [23], [24]. In particular, context monitoring, including demographic background, user impairments, and system-level factors such as end-device hardware and software, is required for assessing reproducibility reliably.

VII. LESSONS LEARNED

A. SUMMARY

In our real-world 5G use case, we have observed the importance of documenting and controlling a variety of factors that contribute to result variability and thus reproducibility. This includes not only hardware setup and configuration but also environmental conditions, testbed-specific calibration, and system loads.

For the Internet measurement use case, we have again emphasized the requirement of careful documentation and control of influence variables like OS, device architecture, and network behavior. We further observed that even virtualization techniques like Docker, replicating the OS, are no guarantee for reproducibility, especially when differences in CPU architecture or OS settings may affect the result outcomes. We have discussed that additional variability from wireless interference, device load, and environmental factors must be accounted for in experimental design. Finally, the integration of the Internet and Internet-based services adds another layer of variance due to network and service variability, making reproducibility even more difficult. However, it is also important to state that reproducibility in this use case strongly depended on the research question and target variables. While most application-layer metrics like download duration or video resolution in DASH were reproducible across measurements and devices, network-layer metrics were much harder to reproduce.

For the third use case, AI/ML networking experiments, we have discussed how data variability and inherent randomness may affect reproducibility and how such experiments are mostly software-based. The major issue with this type of experiment is to achieve transparency. This means that it is essential to document the exact dataset splits, random seeds, and the training process, such that other teams can reuse artifacts or configure their experiments in an identical fashion. The achieved transparency along with a sufficient number of experiment iterations enables reproducibility.

Finally, in the QoE use case, we have discussed how both technical and human factors may affect reproducibility of QoE experiments. We highlighted the trade-off between reproducibility and statistical significance, introduced by the differences between laboratory studies, with lower generalizability and higher reproducibility, and crowdsourcing studies, with higher generalizability and lower experimental control and thus potentially lower reproducibility. Applying the best practices established in crowdsourcing studies, however, might help to improve the reliability of such studies while preserving statistical strength.

Summarizing, it is crucial to understand and address the diverse sources of variability and formulate a proper research question for designing robust reproducible experiments. Thorough documentation of testbed setups, systematic

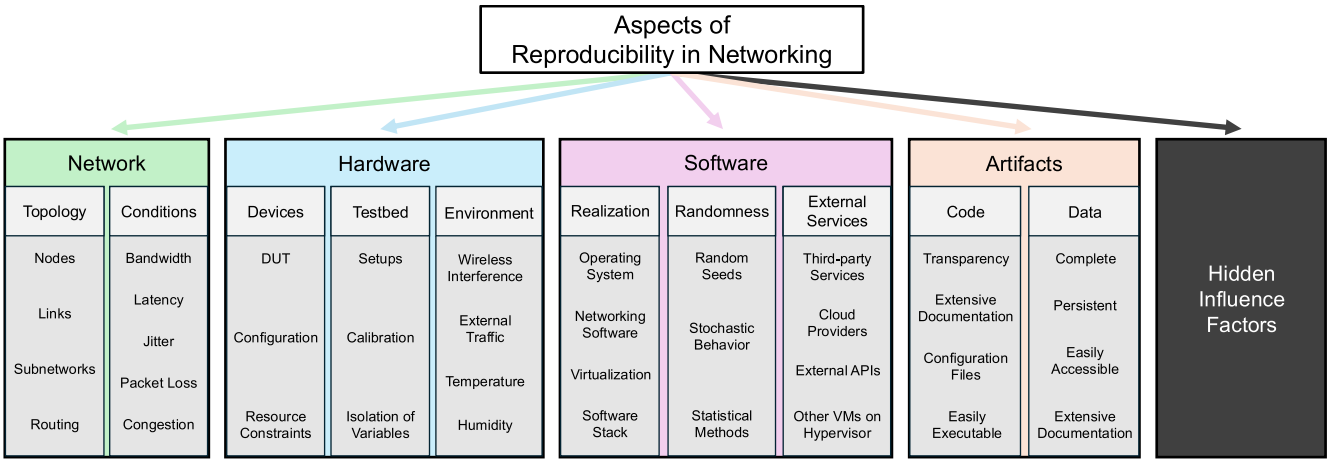


FIGURE 6. Taxonomy for reproducible networking experiments.

variable isolation through means of emulation, and the use of statistical methods to address randomness can reduce the impact of these factors on reproducibility.

B. TAXONOMY AND GUIDELINES

We provide a simple taxonomy for reproducible networking experiments in Figure 6, where we summarize the different aspects relevant for networking experiments along with some examples, which should be kept in mind, when targeting reproducible networking experiments. Notably, we also outline hidden influence factors, capturing unknown or unobserved variables that are not part of the other aspects.

The major cause for irreproducible results is the inherent variability of these components, as shown throughout this work. To reduce this variability and subsequently increase reproducibility, several steps have to be performed. First, it is important to identify the components of high variability directly affecting reproducibility for the considered use case/experiment in dependency of the research question, e.g., physical link or interferences in 5G experiments, closed-box service dependency in video streaming QoE measurements, or hardware randomness in ML experiments. Afterwards, these irreproducible components could be isolated or replaced with reproducible components by utilizing abstracted components based on the given requirements, e.g., replacing the physical link in a 5G network with a radio channel emulation tool. However, such substitutions are not always possible, particularly when experiments include real users or real-world applications. Generally, the variability decreases with increasing abstraction, i.e., the degree of variability is the highest for measuring a real-world system and the lowest for using a fully controllable simulation or model. Finally, documenting every setting, storing experiment artifacts, data, and results, and providing these to other researchers is also fundamental for reproducible networking experiments. It should be further encouraged to provide entire experiment system snapshots. However, as we have observed during our experiments, using tools like Docker may introduce additional challenges, as Docker utilizes the host kernel under the hood.

The proposed guidelines are thus intended as directional guidance rather than providing a solution for every possible scenario. In particular, in networking it is difficult to identify and account for all relevant factors that might affect reproducibility. Our study covers only a limited set of use cases, and generalizing to other contexts may miss important factors influencing the outcome. Thus, the guidelines should be applied carefully with respect to the context and may require additional adjustments for the experimental settings.

C. GENERAL WORKFLOW

Given the broad range of reproducibility experiments in networking, differing for example in objectives, experimental setups, and evaluation metrics, it is impossible to provide a unified workflow covering all possible scenarios. We thus focus on the most common use case: evaluating result reproducibility by replicating the original testbed and conducting statistical comparisons between the measurements obtained from the original and replicated setup. Figure 7 illustrates how the proposed taxonomy guides the workflow of reproducible networking experiments in this context.

Consider an initial experiment conducted by research team A, which produces network measurements X using its testbed. Depending on the research question, an arbitrary function f is applied to X to obtain the final result $\hat{X}$. This function f may represent a simple statistic, e.g., mean or variance, the identity function, i.e., the entire distribution of measurement values, or a more complex model fitted to the data.

We identify three motivations why research team B may aim to reproduce team A's results. Thus, we focus on results reproducibility here. First, team B may question the validity of the results, particularly if they contradict their own hypotheses, and thus want to verify the results'

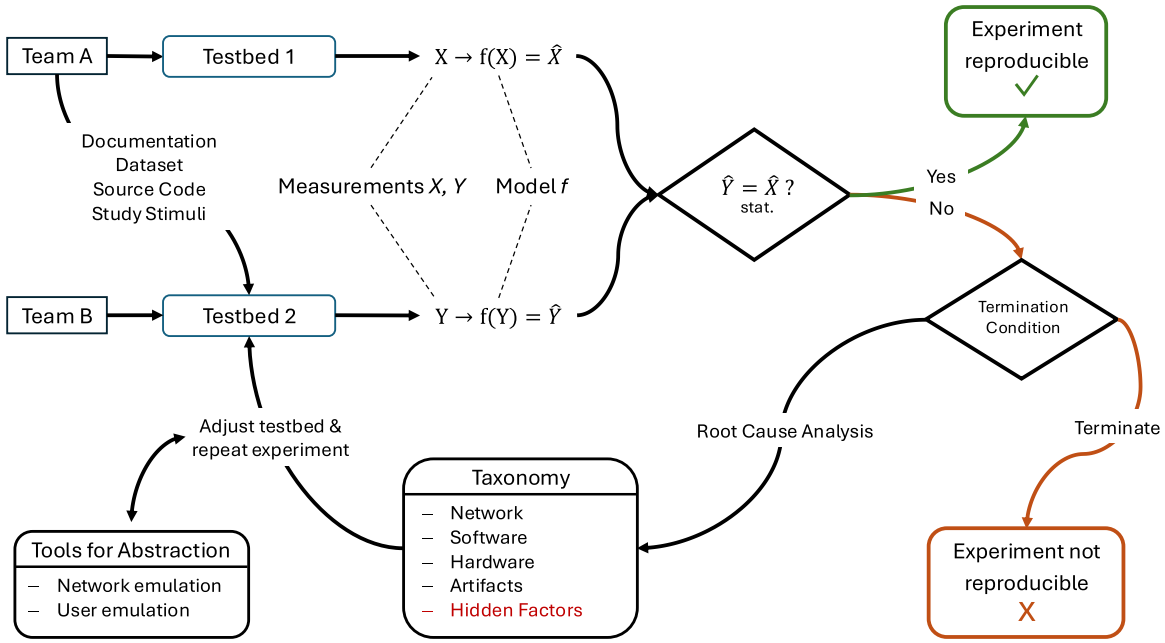


FIGURE 7. Workflow for designing reproducible experiments and integrating the taxonomy.

correctness. Second, team B may be interested in deriving another variable using team A's replicated testbed but must first validate the correctness of their testbed implementation by comparing their results against team A's findings for the same experiment. Third, team B may aim to perform data augmentation to enable large-scale experiments, e.g., machine learning experiments, based on an extended dataset.

Independent of the underlying motivation, the workflow remains the same. Team B uses all available documentation, dataset, source code, and other artifacts to rebuild team A's testbed, and obtain new measurements Y. Analogous to team A, team B applies the same function f to obtain the final result $\hat{Y}$. Reproducibility is then assessed by comparing $\hat{Y}$ with $\hat{X}$ and evaluating the agreement with respect to the research question, either qualitatively or using statistical measures. If simple summary statistics like mean are compared, an acceptable tolerance must be defined to decide whether the results are still sufficiently similar. When comparing full measurement distributions, statistical tests, e.g., Mann-Whitney U, or distance measures, e.g., Wasserstein distance, may be applied. Distance-based approaches again require predefined tolerance, whereas hypothesis tests allow reproducibility to be evaluated via significance level, e.g., α, and corresponding p. Finally, for fitted models, prediction differences must be similarly evaluated against predefined expectations or acceptance criteria. Instead, the appropriate comparison method depends strongly on the specific research question and the form of the reported results.

If the comparison between $\hat{Y}$ and $\hat{X}$ yields test statistics within these predefined acceptance criteria, reproducibility can be considered achieved. Otherwise, team B must revise its testbed and repeat the measurements. To identify the causes of the discrepancy, team B performs a root cause analysis. In this process, team B can consult our taxonomy to systematically investigate potential sources of deviation in their testbed. Specifically, team B can iteratively analyze and adjust components related to network, hardware, software, and experimental artifacts to improve agreement between results and thus reproducibility. This process may include manual or script-based comparisons of testbed configurations, as well as automated analyses to capture explainable variance and configuration-induced differences across experiments. However, fully identifying and controlling all relevant factors that influence reproducibility in networking remains challenging. Our taxonomy thus should be regarded as directional guidance rather than a guarantee for completeness. In particular, unknown or hidden factors may still lead to discrepancies between testbeds. After identifying potential sources of discrepancy, the testbed must be adjusted. This step may involve modifying configurations or replacing components of the testbed using dedicated emulation tools, e.g., to ensure reproducible network conditions or user behavior. To prevent an indefinite repetition of experiments in cases where adjustments fail to yield the desired outcome, a termination criterion is required. Such a criterion may define, for example, a maximum number of reproduction attempts or a predefined resource budget, after which the experiment is considered non-reproducible.

D. TOOLS FOR ABSTRACTION

During testbed refinement, abstraction tools can support improved reproducibility. A variety of such tools already exists to abstract different layers of the networking

experiment stack. Radio channel emulation tools, for example, fix the reproducibility problems caused by random interference. Many such tools exist, e.g., commercial tools like Spirent[5] and Keysight[6] or academic tools like OpenAirLink [25]. If interference is not subject of the evaluation itself, the antennas could be replaced by cables attached directly to the antenna sockets [26]. This way, the experiment can be conducted with real hardware and no random interference involved.

Other tools like Mininet[7] or Linux TC[8] aim to replicate the Quality of Service of a channel, instead of replicating the underlying cause of the service degradation, making things more abstract, but more accessible.

Finally, emulating user behavior to affect network usage is often essential for Internet measurements. Here, tools like Selenium[9] and Appium[10] can provide reliable solutions for consistent behavior emulation.

While various tools exist for specific aspects, there is a lack of standardized tools for conducting a comprehensive root cause analysis of experiments with respect to reproducibility.

VIII. CONCLUSION

Reproducibility is essential for validating and building on scientific research, including networking. While previous works such as Dagstuhl's beginner guide offers general practices for achieving reproducibility in networking experiments, they often overlook specific factors that hinder it. This work addressed those gaps by identifying the limiting factors across four use cases: 5G measurements, local/Internet measurements, AI/ML-based experiments, and QoE experiments. From our analysis, we developed a taxonomy highlighting the most important aspects preventing reproducible research in networking. We derived guidelines and principles to enhance reproducibility in the networking field.

First, research questions must be clearly formulated and the expected evaluation outcomes must be specified. The more precise these definitions, the more reproducible the corresponding experiments tend to be. Second, identifying and documenting sources of randomness specific to the use case is essential prior to designing the experiments. For instance, the influence of Docker on macOS on networking KPIs should be determined and documented in advance. Where possible, unnecessary sources of variability should be minimized, e.g., preferring wired over wireless links when wireless conditions are not part of the research question. Next, detailed documentation, e.g., of a testbed, is fundamental when publishing results, and associated datasets should be thoroughly described and specified. Equally important is the acknowledgment of any limitations or challenges encountered in the original setup, as it helps prevent future researchers from obtaining the same misleading results. Finally, in some cases, introducing deliberate randomness might strengthen the statistical validity of findings, e.g., increasing the number of participants in human-related research or using multiple random seeds in AI-based experiments.

Accounting for all sources of variability is often impractical due to the inherent complexity of networks. Our taxonomy and guidelines, however, assist researchers in designing more reproducible experiments, fostering reproducible networking research. The taxonomy is intended to synthesize insights, highlight potential pitfalls, and provide guidance rather than serve as a fully validated framework. Still, a formal validation in future work could strengthen its authority. Future research could also focus on developing a more structured framework for balancing reproducibility and statistical power in decision-making, as well as in identifying specific metrics for reproducibility assessment.

[5]https://www.spirent.com/assets/u/spirent-network-emulator-flyer

[6]https://www.keysight.com/de/de/products/network-test/network-test-hardware/network-emulator-ii.html

[7]https://mininet.org/

[8]https://man7.org/linux/man-pages/man8/tc.8.html

[9]https://www.selenium.dev/

[10]https://appium.io/docs/en/latest/

ACKNOWLEDGMENT

This work contributes to SLICES-DE (https://slices-de.org/). The authors alone are responsible for the content.

REFERENCES

[1] V. Bajpai, A. Brunstrom, 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," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 49, no. 1, pp. 24–30, Feb. 2019.

[2] F. Rezabek, M. Bosk, T. Paul, K. Holzinger, S. Gallenmüller, A. Gonzalez, A. Kane, F. Fons, Z. Haigang, G. Carle, and J. Ott, "EnGINE: Flexible research infrastructure for reliable and scalable time sensitive networks," *J. Netw. Syst. Manage.*, vol. 30, no. 4, p. 74, Oct. 2022.

[3] *Artifact Review and Badging Version 1.1*. Accessed: Jan. 15, 2024. [Online]. Available: https://www.acm.org/publications/policies/artifact-review-and-badging-current

[4] *Artifact Review and Badging Version 1.0*. Accessed: Jan. 15, 2024. [Online]. Available: https://www.acm.org/publications/policies/artifact-review-badging

[5] *Reproducibility and Replicability in Science*, National Academies Press, Washington, DC, USA, 2019.

[6] O. E. Gundersen and S. Kjensmo, "State of the art: Reproducibility in artificial intelligence," in *Proc. AAAI Conf. Artif. Intell.*, 2018, vol. 32, no. 1, pp. 1644–1651.

[7] N. Demir, M. Große-Kampmann, T. Urban, C. Wressnegger, T. Holz, and N. Pohlmann, "Reproducibility and replicability of Web measurement studies," in *Proc. ACM Web Conf.*, Apr. 2022, pp. 533–544.

[8] P. Cimiano, C. Pietsch, and C. Wiljes, "Studies in analytical reproducibility: The conquaire project," Universitätsbibliothek Bielefeld, Bielefeld, Germany, Tech. Rep., 2021. [Online]. Available: https://noah.nrw/ubbihs/content/titleinfo/5141199

[9] P. Brunisholz, E. Dublé, F. Rousseau, and A. Duda, "WalT: A reproducible testbed for reproducible network experiments," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2016, pp. 146–151.

[10] S. Gallenmüller, D. Scholz, H. Stubbe, and G. Carle, "The pos framework: A methodology and toolchain for reproducible network experiments," in *Proc. 17th Int. Conf. Emerg. Netw. EXperiments Technol.*, Dec. 2021, pp. 259–266.

[11] M. Bosk, F. Rezabek, K. Holzinger, A. G. Marino, A. A. Kane, F. Fons, J. Ott, and G. Carle, "Methodology and infrastructure for TSN-based reproducible network experiments," *IEEE Access*, vol. 10, pp. 109203–109239, 2022.

[12] N. Demir, J. Hörnemann, M. Große-Kampmann, T. Urban, N. Pohlmann, T. Holz, and C. Wressnegger, "On the similarity of web measurements under different experimental setups," in *Proc. 2023 ACM Internet Meas. Conf.*, 2023, pp. 356–369.

[13] F. Hantke, S. Calzavara, M. Wilhelm, A. Rabitti, and B. Stock, "You call this archaeology? Evaluating Web archives for reproducible Web security measurements," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2023, pp. 3168–3182.
[14] S. Raffeck, S. G. Grøsvik, S. Lange, T. Hossfeld, T. Zinner, and S. Geissler, "Parameterizing 5G new radio: A comparative measurement study on throughput and delay," in *Proc. 20th Int. Conf. Netw. Service Manage. (CNSM)*, Oct. 2024, pp. 1–7.
[15] *Networking on Docker Desktop*. Accessed: Dec. 18, 2025. [Online]. Available: https://docs.docker.com/desktop/features/networking/
[16] M. Seufert, B. Zeidler, F. Wamser, T. Karagkioules, D. Tsilimantos, F. Loh, P. Tran-Gia, and S. Valentin, "A wrapper for automatic measurements with YouTube's native Android app," in *Proc. Netw. Traffic Meas. Anal. Conf. (TMA)*, Jun. 2018, pp. 1–8.
[17] *Tf.keras.utils.set_random_seed*. Accessed: Jun. 30, 2025. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/utils/set_random_seed
[18] *Reproducibility*. Accessed: Dec. 18, 2025. [Online]. Available: https://github.com/pytorch/pytorch/blob/main/docs/source/notes/randomness.rst
[19] *TF.config.experimental.enable_op_determinism*. Accessed: Dec. 18, 2025. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/config/experimental/enable_op_determinism
[20] T. Hoßfeld, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia, and R. Schatz, "Quantification of YouTube QoE via crowdsourcing," in *Proc. IEEE Int. Symp. Multimedia*, Dec. 2011, pp. 494–499.
[21] N. Wehner, N. Mertinat, M. Seufert, and T. Hoßfeld, "Studying the impact of the content selection method on the video QoE on mobile devices," in *Proc. 12th Int. Conf. Quality Multimedia Exper. (QoMEX)*, May 2020, pp. 1–4.
[22] S. Egger-Lampl, J. Redi, T. Hoßfeld, M. Hirth, S. Möller, B. Naderi, C. Keimel, and D. Saupe, "Crowdsourcing quality of experience experiments," in *Evaluation in the Crowd. Crowdsourcing and Human-Centered Experiments*. Cham, Switzerland: Springer, 2017, pp. 154–190.
[23] T. Hoßfeld, M. Hirth, J. Redi, F. Mazza, P. Korshunov, B. Naderi, M. Seufert, B. Gardlo, S. Egger, and C. Keimel, "Best practices and recommendations for crowdsourced QoE—Lessons learned from the Qualinet task Force crowdsourcing," COST Action IC1003 Eur. Netw. Quality Exper. Multimedia Syst. Services (QUALINET), 2014. [Online]. Available: https://hal.science/hal-01078761
[24] T. Hossfeld, C. Keimel, M. Hirth, B. Gardlo, J. Habigt, K. Diepold, and P. Tran-Gia, "Best practices for QoE crowdtesting: QoE assessment with crowdsourcing," *IEEE Trans. Multimedia*, vol. 16, no. 2, pp. 541–558, Feb. 2014.
[25] Y. Deshpande, X. Wang, and W. Kellerer, "Openairlink: Reproducible wireless channel emulation using software defined radios," 2024, *arXiv:2404.09660*.
[26] *Building a Wireless Testbed. With Wires!*. Accessed: Jul. 24, 2025. [Online]. Available: https://blog.tohojo.dk/2017/11/building-a-wireless-testbed-with-wires.html

NIKOLAS WEHNER received the master's degree in computer science, in 2017, and the Ph.D. degree, in 2025. He is currently a Postdoctoral Researcher with the University of Würzburg, Germany. Before joining the Center for Technology Experience, AIT Austrian Institute of Technology, Vienna, Austria, as a Research Engineer. In 2019, he joined the Chair of Communication Networks, University of Würzburg, as a Research Assistant. His research interests include user-centric communication networks, focusing on the QoE of multimedia applications and the application of machine learning.

ALEXEJ GRIGORJEW received the M.Sc. degree in computer science from the University of Würzburg, Würzburg, Germany, in 2016, where he is currently pursuing the Ph.D. degree with the Chair of Communication Networks. His early research interests include SDN, NFV, and programmable data planes. Since 2018, his main focus has been on low-latency networking and TSN, with an emphasis on distributed latency-bound analysis and network configuration.

SIMON RAFFECK received the master's degree from the University of Würzburg, Germany, in 2022, where he is currently pursuing the Ph.D. degree with the Chair of Communication Networks, where he is part of the Cloud Applications and Networks Research Group. His current research is focused on the analytical and simulative performance evaluation and optimization of next-generation communication technologies, including 5G, 6G, the Internet of Things, and TSN, as well as the energy efficiency of wireless communication and systems.

MICHAEL GOEDICKE received the Ph.D. degree in computer science from the Technical University of Dortmund, in 1985, on specification languages for embedded systems, and the Habilitation degree, in 1993, on the topic of specification of software components. He then conducted research in the areas of the specification of software architectures and the description of software components. He has been a Professor of practical computer science/specification of software systems with the University of Duisburg–Essen, since 1994. His area of research interests include software architecture and applications, specifically in the area of eAssessment systems and research data management in computer science.

TOBIAS HOßFELD (Senior Member, IEEE) received the Ph.D. degree, in 2009, and his professorial thesis (Habilitation), in 2013. From 2014 to 2018, he was the Head of the Chair of Modeling of Adaptive Systems, University of Duisburg–Essen. He has been a Professor with the Chair of Communication Networks, University of Würzburg, Germany, since 2018. He is a member of the editorial board of IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, *Quality and User Experience* (Springer), and ACM SIGMM Records, and the Elected Chairperson of the VDE ITG Expert Group "Communication Networks and Systems" within the German Society of Information Technology (ITG). Among others, he received the Fred W. Ellersick Prize 2013 (IEEE Communications Society) for the generic relationship between QoS and QoE and the VDE ITG Award 2024 for comparing the scalability of communication networks and systems.