Usage of Path Property Emulation Tools

Julien Schiffer, Stefan Lachnit*, Sebastian Gallenmüller*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: schiffer.ju@tum.de, lachnit@net.in.tum.de, gallenmu@net.in.tum.de

***Abstract*—There are many different emulation tools, some with similar but different functions and goals. The most common ones are Mininet and NetEm. They are often used to test and validate the work of researchers. This includes new algorithms or programs. However, it is not always clear why a particular emulation tool was used. The following paper presents and categorizes the use of those tools. Meanwhile, it becomes clear that current emulation tools have technical limitations and are sometimes reaching their limits, which is why there will be even more powerful tools in the future.**

***Index Terms*—Network Emulation, Path property, Emulation tools, Mininet, Netem**

1. Introduction

As the Internet grows, the corresponding programs and requirements become more complex. It is not only important that applications work at all, but speed also plays a significant role. There are a lot of devices with different hardware on which the programs should still be usable quickly. Therefore, tests must be carried out to check how programs behave with other parameters, such as high latency and bandwidth. The aim is to recreate real network conditions. Entire virtual networks can also be created to carry out tests.

The use of such tools offers many advantages compared to testing on real networks. The most significant advantage is efficiency. Networks can be set up and used much faster. Moreover, the costs of real hardware are eliminated. Only with large and complex networks can the performance decrease, meaning the test results are not 100 % accurate. Nevertheless, the cost benefits outweigh the disadvantages in most cases, which is why emulation tools are used so frequently.

In the following paper, the emulation tools are presented and compared with each other, documenting their use over the last five years.

2. Emulation Tools

2.1. Function

Emulation tools make it possible to recreate real hardware or software environments to test their behavior in a controlled environment. Realistic networks or systems are simulated without the need for the underlying hardware. The tool must be as similar as possible to the original system. Network emulation tools, in particular, allow certain conditions to be emulated by changing individual parameters. This allows for better evaluation of experiments.

For example, it is possible to artificially increase the latency and see how the same application works on less powerful devices. Other limitations, such as low bandwidth or high packet loss, can also be emulated. It is also possible to emulate real networks, which function like real networks.

2.2. Most Common Emulation Tools

Mininet is by far the most frequently used tool in the papers. It can mimic a real network by creating virtual hosts, switches, controllers, and connections [1]. That is why it is primarily used when researchers want to focus on Software-defined Networks (SDN). It can also simulate targeted network failures. The simulations result in data sets that can be used to train models [2]. Because only one device is required, Mininet is very cost-efficient. However, Mininet-based networks cannot exceed the CPU or bandwidth available on a single server. This leads to bottlenecks when too many network components are emulated, resulting in increased latency, packet loss, or inaccurate bandwidths.

Another well-known tool is GNS3 [3]. Compared to Mininet, GNS3 is a more robust network emulator that is mainly used for simulating traditional networks. It can connect real network devices, such as Cisco routers and switches, to virtual machines (such as Linux servers). GNS3's emulation is more realistic because it uses official operating system images (such as Cisco IOS). However, these are often licensed, so it can be challenging to get hold of them. In addition, GNS3 is more resource-intensive due to emulating real devices, which is why it is slow on more extensive networks.

Emulab [4], meanwhile, connects physical and virtual networks. Real hardware (physical devices) and also virtual machines are used. Emulab enables the use of many physical machines in a distributed environment, which increases scalability.

Containernet is an extension of Mininet [5]. In addition to Mininet's functions, it enables the use of Docker containers as hosts, which allows the use of real services (e.g., microservices).

NetEm [6], on the other hand, focuses on changing specific parameters. It is possible to add latency, simulate packet loss and jitter, limit bandwidth, and much more. This makes the tool well-suited for performance testing and software optimization.

 doi: 10.2313/NET-2025-05-1_08

TABLE 1: Usage of Path Property Tools in Paper

	overall	Mininet	NetEm	GNS3	EmuLab	other
2020	10	5	1	0	3	1
2021	12	7	1	1	0	3
2022	6	5	1	0	0	0
2023	5	2	1	1	0	1
2024	6	2	2	0	0	2
sum	39	21	6	2	3	7

Dummynet [7] is similar to NetEm; it was developed only on FreeBSD, but nowadays, it is also usable on MacOS or Linux. NetEm, on the other hand, is based on Linux and is only installable on a Linux-based operating system. Dummynet focuses more on traffic shaping, which is why it is more flexible and efficient in those areas. However, that also needs a deeper configuration and more system resources. Both are applied on the network interface and act as an intermediate layer between devices and the network.

2.3. Installation

Setting up Mininet is easy. The first step is to install a Mininet VM image and then open it in a virtualization system(Quelle). This setup is quick because the Mininet VM has a pre-built environment that is ready to be used. NetEm, on the other hand, can only be installed on a Linux-based operating system such as Ubuntu because it is not a standalone program(Quelle). The corresponding websites provide 'codes' that make it possible to use those tools. For example, in NetEm, 10% packet loss is simulated with the code: `sudo tc qdisc change dev eth0 root netem loss 10%`.

Comparable programs can be installed on FreeBSD (Dummynet [7]).

3. Usage in papers

In the following, we reviewed the papers of ACM SIGCOMM and ACM CoNEXT and found all the papers in which path property emulation tools are used. Table 1 shows the use of the various tools from 2020 to 2024.

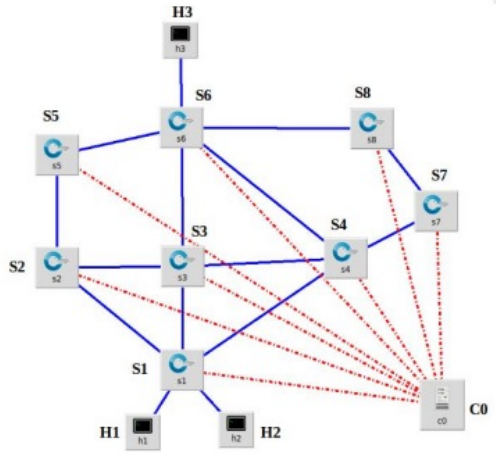


Figure 1: An Example Network Topology

3.1. Mininet

Mininet was used 21 times, representing about half of all papers using emulation tools. In references [8], the authors created an automatic Mininet topology generator. This creates "an evaluation testbed for any kind of measurements that require a ground-truth dataset" [8]. A ground truth dataset is a reference standard for evaluating models or algorithms. The accurate and verified dataset information must often be created manually, which the authors want to automate in this paper. The paper focuses primarily on Resource Public Key Infrastructure(RPKI) measurements, but they want to make the approach possible for all measurements requiring a ground truth dataset. The first step is to collect the required data from public sources to create a directed graph. Since Mininet is not powerful enough to simulate the entire topology, the framework must reduce the graph to approximately 4000 nodes while maintaining important properties such as the degree distribution of the nodes. The abstracted graph is then translated into a Mininet configuration file, which, in this case, generates a realistic RPKI deployment scenario. The developed program allows filtering on a self-selected set of nodes in the Mininet topology, which can create a ground truth dataset.

In Paper [9], an algorithm is created that finds the ideal path between two nodes in a network. It is not about the length of the path but about better traffic load distribution. Figure 1 shows a network created in Mininet, where different hosts are connected to a controller via many switches. Data is sent from the hosts with different bandwidths to check whether their algorithm will find the best path. There are three scenarios: once both hosts send with less than 50% of link bandwidth, once one host sends with more than 50%, and the last time both send the data with more than 50% bandwidth. In all three scenarios, the algorithm prevented overload, proving the quality of the algorithm.

It is also possible to train data sets with Mininet, done in [2], [10]. In the paper [2], datasets of network failures were simulated in Mininet to train decision-tree-based models. For this purpose, random traffic was emulated in selected topologies, and then failure was injected by manually setting the status of links and nodes to failure.

Sometimes, a network consisting of a client and a server, both dual-stacks, is emulated [11]. Then, the latency and bandwidth of the IPv4 and IPv6 paths can be adjusted.

It is also possible that the features of Mininet are not sufficient. For example, in Paper [12], IPMininet, an extension of Mininet that supports SRv6, is used. SRv6 is a modern and flexible routing technology that directly integrates the segment routing principle into the IPv6 protocol. This makes routing more efficient because packets carry explicit path information, and the behavior of networks is made programmable. IPMininet also allows the evaluation of network conditions, such as packet loss, by emulating network loss. This allows the authors to test their SRv6 plugin.

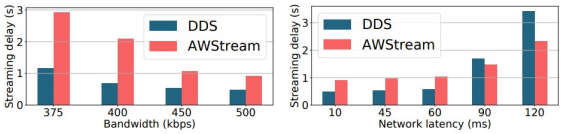
Primarily, Mininet is used to simulate networks with routers, switches, and hosts. Often, the bandwidth, latency, and other parameters are actively set. In Paper [13],

 doi: 10.2313/NET-2025-05-1_08

routers are used whose links have a bandwidth limit of 1000 Mbit s^{-1} and a constant delay of 2 ms.

The other papers also briefly address the use of Mininet to emulate topologies [14]–[25] but do not elaborate further.

In summary, Mininet can be used flexibly. Fundamentally, it is used to emulate networks on which tests are carried out. These include testing network performance or the functionality of one's algorithm. Sometimes, data sets are created from the information obtained, which are used to train models.



(a) Bandwidth vs. streaming delay **(b) Latency vs. streaming delay**

Figure 2: The response delay of AWStream and DDS with different network bandwidth and latency

3.2. NetEm

The use of NetEm in papers has been entirely one-sided. It is mainly described that parameters such as bandwidth and latency in networks have been varied [13], [26]. In paper [26], this was done to test the performance of DDS, a streaming technique. It is a model developed for video streams from cameras to be sent to servers, which will be processed using deep neural networks (DNNs). DNNs are an artificial neural network used to solve complex machine-learning tasks. In Figure 2, the streaming delay of DDS compared to an already established system (AWSStream), with different bandwidth and latency, is shown. The exact comparison is possible through the emulation in NetEm.

Other parameters, such as packet loss and long delay (due to a high round-trip time), can also be emulated [27], [28]. A 5G setup, a combination of increased jitter (fluctuations in the delay), lower bandwidth, and delays, and an LTE-M model were also simulated, which usually offers very low latency and high bandwidth.

In Paper [29], the bottleneck link's speed and the bottleneck buffer's size were configured to gain control over the network. This affects the network performance so that the authors can evaluate the response to network congestion and packet loss.

In summary, the users of this emulation tool are satisfied with the tool. There are limitations when the load is too high (for example, too much jitter), but this was not a problem in the papers mentioned.

3.3. Emulab

In Reference [30], extensive simulations are performed with Emulab. Different network topologies are emulated, with different conditions, to demonstrate the performance of "MPCC, a high-performance multipath congestion control architecture" [30]. It was explicitly declared that "[u]nless stated otherwise all link latencies, bandwidths, and buffer sizes are 30 ms, 100 Mbit s^{-1} and 375 kB (the Bandwidth-Delay Product), respectively." [30] All values are detailed and explained with potential limitations for other values. It is noted that the experiments in Emulab cannot be 100% transferred to real networks. Therefore, live experiments were also carried out in which files were downloaded from various locations in the cloud. This also implies that the tests in Emulab alone cannot provide complete information about the tool in practice.

The Paper [31] also emulates parameters such as latency, focusing on dynamic changes in network conditions.

It should be noted that papers that use Emulab conduct extensive and detailed experiments [17]. Emulab is designed for larger network emulations, increasing the complexity and time needed to understand the program.

3.4. GNS3

In Paper [32], [33], Gns3 was used similarly to Mininet to create virtual networks and test algorithms. Paper [32] is about Snowcap, which requires GNS3 to function. A detailed guide is provided explaining how to use Snowcap, one of the requirements being GNS3.

3.5. Other Tools

There is one paper that uses Dummynet [34]. Again, realistic network conditions were simulated, which were intended to represent mobile networks. These are known for fluctuating latencies, achieved in Dummynet by constantly adjusting the queue delays with target jitter values. The method caused latency fluctuations without letting packets arrive in the wrong order, which more realistically simulates mobile networks. Dummynet was used here because it is more focused on traffic shaping than NetEm, offering greater flexibility in emulation. However, the paper mentions that Dummynet is limited by a maximum queue size of 100, which does not allow for entirely flexible emulation.

Nanonet is an emulation tool conceptually based on Mininet and primarily aimed at segment routing experiments [35]. Segment routing is routing traffic in networks on predefined routers instead of making a new decision at each router. In Paper [35], they simulate network nodes connected to each other for a realistic simulation of network behavior. Nanonet then calculates the shortest path between the nodes. An even distribution of traffic on the paths was ensured to measure the maximum link utilization with a specific instance. This allowed testing and comparing different approaches for optimizing link weights.

Another extension of Mininet is G2-Mininet, which analyzes the Quantitative Theory of Bottleneck Structures (QTBS). QTBS is a mathematical theory developed to analyze and optimize communication networks. The paper simulated various network topologies (fat trees, folded clos, and Dragonfly) to test QTBS. More than 600 networks were simulated over more than 800 hours to confirm the correctness of the model.

In Paper [36], SimBricks, a Network System Evaluation with Modular Simulation, was introduced with ns-3 integrated. Ns-3 [37] is also an emulation tool that can process and synchronize packets using the Ethernet

network interface. SimBrick uses ns-3 to simulate network layers and connections between virtual and physical network topologies.

The paper [5] takes advantage of the fact that Containernet is based on containers that form a network of interconnected nodes. Such a network is created with each container running a KIRA routing server that provides IPv6 connectivity and some containers running additional 5G core network functions. This is part of the KIRA routing architecture intended to enable autonomous and fault-tolerant network control. Containernet is mainly used to emulate the network topology and test node failures.

A tool yet to be mentioned is BESS. It allows the user to control network properties such as latency at a more detailed level than other tools such as NetEm [38]. This enables fine-grained control of network traffic. In the paper [38], it is used "to control the access link speed, queue size, and add delay to ingress and egress packets" [38] of a switch. In addition, BESS can measure important data such as queue occupancy and packet loss, which enables more precise analysis.

The last paper introduces Klonet [39]. It is a new network emulation platform that was created for educational purposes. It is criticized that current emulation tools are inadequate for integrating network hardware due to insufficient scalability and other factors. The paper also creates a table with the tools currently in use and shows factors such as hardware support, container support, and VM support.

3.6. Summary

	Mininet	NetEm	GNS3	EmuLab
Function	Simulation of Software-Defined Networks (SDN)	Simulation of latency, packet loss, etc.	Simulation of physical and virtual devices	Testing and experimenting with real networks
Architecture	Virtual Switches, Hosts and Controllers	Works as part of Linux Traffic Control	Virtual machines, physical devices and switches	Hardware and software testbed
Scalability	Good for small to medium networks	Depending on the hardware	Very good for small to large networks	High scalability (both small and large networks)
Realism	High proximity to SDN-based networks	Not an exact replica, only simulates properties	Close to real networks	Very high realism through real hardware

Figure 3: Comparison of Path Property Emulation Tools

Many different emulation tools are used for different reasons. In Figure 3, the most commonly used tools are compared. For example, Mininet was often used in the papers, mainly to emulate small and medium-sized networks. NetEm was used to emulate network properties but was only mentioned briefly. Although GNS3 is more flexible in the size of the emulated networks, the tool was mostly only mentioned in passing. Papers that use Emulab have mainly carried out very extensive and detailed experiments that comprise several pages of the paper. The complexity of the tool can explain this. For example, while the functions of NetEm are limited to emulating parameters on one host, Emulab can emulate large networks with complex properties on several hosts, switches, and routers.

Nevertheless, the current tools are not perfect. Sometimes, the current applications do not meet the requirements. Very few tools can simulate specific hardware properties or modern technologies such as containers. Also, with large, realistic networks, many tools reach their limits. This has led to tools such as Klonet and Containernet, which are new emulation tools with more options that can be used in a broader variety of ways.

4. Conclusion

Path property emulation tools play a crucial role in network research and development. It is possible to precisely simulate network properties such as latency, bandwidth, loss, and jitter. Depending on the requirements, choosing the right emulation tool can be cost and time-efficient, mainly when a lot of data has to be processed.

However, since the established tools are imperfect, their results cannot always be 100% transferred to the real world. With the rapid development of the Internet, there will be even more powerful and diverse future tools capable of more. Technologies such as artificial intelligence also mean a lot is still possible in this area, making it possible that currently established tools will become outdated and no longer be used.

Nevertheless, the current tools make an essential contribution to the development of modern networks. Even if they are imperfect, most of the tools have been in use for several years and will continue to be used in various ways and on a wide scale.

References

[1] Mininet, https://mininet.org/overview/, 19.10.2024.

[2] X. Zuo, Q. Li, J. Xiao, D. Zhao, and J. Yong, “Drift-bottle: a lightweight and distributed approach to failure localization in general networks,” in *Conference: CoNEXT ’22: The 18th International Conference on emerging Networking EXperiments and Technologies*, 11 2022, pp. 337–348.

[3] GNS3, https://www.gns3.com/, 19.10.2024.

[4] Emulab, https://www.emulab.net/portal/frontpage.php, 19.10.2024.

[5] P. Seehofer, H. Mahrt, R. Bless, and M. Zitterbart, “Demo: Enabling autonomic network infrastructures with kira,” in *Conference: ACM SIGCOMM ’23: ACM SIGCOMM 2023 Conference*, 09 2023, pp. 1165–1167.

[6] NetEm, https://man7.org/linux/man-pages/man8/tc-netem.8.html#OPTIONS, 19.10.2024.

[7] DummyNet, https://cs.baylor.edu/~donahoo/tools/dummy/tutorial.htm, 19.10.2024.

[8] N. Rodday, R. Baaren, L. Hendriks, R. Rijswijk-Deij, A. Pras, and G. Dreo, “Evaluating rpki rov identification methodologies in automatically generated mininet topologies,” in *Conference: CoNEXT ’20: The 16th International Conference on emerging Networking EXperiments and Technologies*, 11 2020, pp. 530–531.

[9] K. Abiram and T. Kathiravelu, “Congestion avoidance in data communication networks using software defined networking,” in *Conference: CoNEXT ’21: The 17th International Conference on emerging Networking EXperiments and Technologies*, 12 2021, pp. 463–464.

[10] H. Mostafaei, S. Miri, and S. Schmid, “Poster: Reactnet: Self-adjusting architecture for networked systems,” in *Conference: 17th International Conference on emerging Networking EXperiments and Technologies (CoNEXT 2021 Posters)*, 12 2021.

[11] F. Rochet, E. Assogba, M. Piraux, K. Edeline, B. Donnet, and O. Bonaventure, "Tcpls: modern transport services with tcp and tls," in *Conference: CoNEXT '21: The 17th International Conference on emerging Networking EXperiments and Technologies*, 12 2021, pp. 45–59.

[12] L. Navarre, F. Michel, and O. Bonaventure, "Srv6-fec: bringing forward erasure correction to ipv6 segment routing," in *Conference: SIGCOMM '21: ACM SIGCOMM 2021 Conference*, 08 2021, pp. 45–47.

[13] J. Zhang, C. Zeng, H. Zhang, S. Hu, and K. Chen, "Liteflow: towards high-performance adaptive neural networks for kernel datapath," in *Conference: SIGCOMM '22: ACM SIGCOMM 2022 Conference*, 08 2022, pp. 414–427.

[14] T. Jepsen, A. Fattaholmanan, M. Moshref, N. Foster, A. Carzaniga, and R. Soulé, "Forwarding and routing with packet subscriptions," in *Conference: CoNEXT '20: The 16th International Conference on emerging Networking EXperiments and Technologies)*, 11 2020, pp. 282–294.

[15] A. Sacco, F. Esposito, and G. Marchetto, "A distributed reinforcement learning approach for energy and congestion-aware edge networks," in *Conference: CoNEXT '20: The 16th International Conference on emerging Networking EXperiments and Technologies*, 11 2020, pp. 546–547.

[16] H. Nagda, R. Nagda, I. Pedisich, N. Sultana, and B. Loo, "Fdp: a teaching and demo platform for sdn," in *Conference: CoNEXT '20: The 16th International Conference on emerging Networking EXperiments and Technologies*, 11 2020, pp. 524–525.

[17] D. Senf, H. Shulman, and M. Waidner, "Performance penalties of resilient sdn infrastructures," in *Conference: CoNEXT '20: The 16th International Conference on emerging Networking EXperiments and Technologies*, 11 2020, pp. 528–529.

[18] P. Bol, R. Lunardi, B. B. França, and W. Cordeiro, "Modular switch deployment in programmable forwarding planes with switch (de)composer," in *Conference: SIGCOMM '21: ACM SIGCOMM 2021 Conference*, 08 2021, pp. 30–32.

[19] D. Guo, S. Chen, K. Gao, Q. Xiang, Y. Zhang, and Y. Yang, "Flash: fast, consistent data plane verification for large-scale network settings," in *Conference: SIGCOMM '22: ACM SIGCOMM 2022 Conference*, 08 2022, pp. 314–335.

[20] S. Sagkriotis and D. Pezaros, "Accelerating kubernetes with in-network caching," in *Conference: SIGCOMM '22: ACM SIGCOMM 2022 Conference*, 10 2022, pp. 40–42.

[21] S. Renganathan, B. Rubin, H. Kim, P. Ventre, C. Cascone, D. Moro, C. Chan, N. McKeown, and N. Foster, "Hydra: Effective runtime network verification," in *Conference: ACM SIGCOMM '23: ACM SIGCOMM 2023 Conference*, 09 2023, pp. 182–194.

[22] K. Namjoshi, S. Gheissi, and K. Sabnani, "Algorithms for in-place, consistent network update," in *Conference: ACM SIGCOMM '24: ACM SIGCOMM 2024 Conference*, 08 2024, pp. 244–257.

[23] C. Jiang, Z. Li, S. Rao, and M. Tawarmalani, "Flexile: meeting bandwidth objectives almost always," in *Conference: CoNEXT '22: The 18th International Conference on emerging Networking EXperiments and Technologies*, 11 2022, pp. 110–125.

[24] L. Brown, A. Gran Alcoz, F. Cangialosi, A. Narayan, M. Alizadeh, H. Balakrishnan, E. Friedman, E. Katz-Bassett, A. Krishnamurthy, M. Schapira, and S. Shenker, "Principles for internet congestion management," in *Conference: ACM SIGCOMM '24: ACM SIGCOMM 2024 Conference*, 08 2024, pp. 166–180.

[25] J. Yen, T. Lévai, Q. Ye, X. Ren, R. Govindan, and B. Raghavan, "Semi-automated protocol disambiguation and code generation," in *Conference: SIGCOMM '21: ACM SIGCOMM 2021 Conference*, 08 2021, pp. 272–286.

[26] T. John, P. Vaere, C. Schutijser, A. Perrig, and D. Hausheer, "Linc: low-cost inter-domain connectivity for industrial systems," in *Conference: SIGCOMM '21: ACM SIGCOMM 2021 Conference*, 08 2021, pp. 68–70.

[27] M. Sosnowski, F. Wiedner, E. Hauser, L. Steger, D. Schoinianakis, S. Gallenmüller, and G. Carle, "The performance of post-quantum tls 1.3," in *Conference: CoNEXT 2023: The 19th International Conference on emerging Networking EXperiments and Technologies*, 12 2023, pp. 19–27.

[28] A. Tahir, P. Goyal, I. Marinos, M. Evans, and R. Mittal, "Efficient policy-rich rate enforcement with phantom queues," in *Conference: ACM SIGCOMM '24: ACM SIGCOMM 2024 Conference*, 08 2024, pp. 1000–1013.

[29] M. Arghavani, H. Zhang, D. Eyers, and A. Arghavani, "Suss: Improving tcp performance by speeding up slow-start," in *Conference: ACM SIGCOMM '24: ACM SIGCOMM 2024 Conference*, 08 2024, pp. 151–165.

[30] T. Gilad, N. Rozen-Schiff, P. Godfrey, C. Raiciu, and M. Schapira, "Mpcc: online learning multipath transport," in *Conference: CoNEXT '20: The 16th International Conference on emerging Networking EXperiments and Technologies*, 11 2020, pp. 121–135.

[31] T. Meng, N. Rozen-Schiff, P. Godfrey, and M. Schapira, "Pcc proteus: Scavenger transport and beyond," in *Conference: SIGCOMM '20: Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 07 2020, pp. 615–631.

[32] T. Schneider, R. Birkner, and L. Vanbever, "Snowcap: synthesizing network-wide configuration updates," in *Conference: SIGCOMM '21: ACM SIGCOMM 2021 Conference*, 08 2021, pp. 33–49.

[33] M. Brown, A. Fogel, D. Halperin, V. Heorhiadi, R. Mahajan, and T. Millstein, "Lessons from the evolution of the batfish configuration analysis tool," in *Conference: ACM SIGCOMM '23: ACM SIGCOMM 2023 Conference*, 09 2023, pp. 122–135.

[34] N. Agarwal, M. Varvello, A. Aucinas, F. Bustamante, and R. Netravali, "Mind the delay: the adverse effects of delay-based tcp on http," in *Conference: CoNEXT '20: The 16th International Conference on emerging Networking EXperiments and Technologies*, 11 2020, pp. 364–370.

[35] M. Parham, T. Fenz, N. Süss, K.-T. Foerster, and S. Schmid, "Traffic engineering with joint link weight and segment optimization," in *CoNEXT '21: Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies*, 12 2021, pp. 313–327.

[36] H. Li, J. Li, and A. Kaufmann, "Simbricks: end-to-end network system evaluation with modular simulation," in *Conference: SIGCOMM '22: ACM SIGCOMM 2022 Conference*, 08 2022, pp. 380–396.

[37] ns 3, https://www.nsnam.org/, 19.10.2024.

[38] A. Philip, R. Athapathu, R. Ware, F. Mkocheko, A. Schlomer, M. Shou, Z. Meng, S. Seshan, and J. Sherry, "Prudentia: Findings of an internet fairness watchdog," in *Conference: ACM SIGCOMM '24: ACM SIGCOMM 2024 Conference*, 08 2024, pp. 506–520.

[39] J. Guo, D. Wu, C. Ma, Y. Hongfang, G. Sun, L. Luo, Y. Xu, and N. Zhang, "Poster: A hybrid virtual-real emulation platform for computer network education," in *Conference: ACM SIGCOMM Posters and Demos '24: ACM SIGCOMM 2024 Conference: Posters and Demos*, 08 2024, pp. 45–47.