Using Cisco VIRL and GNS3 to Improve the Scale-out of Large Virtual Network Testbeds in Higher Education

Sven Reißmann*, Sebastian Rieger†, Christoph Seifert†, Christian Pape†
*Datacenter
Fulda University of Applied Sciences, Fulda, Germany
Email: sven.reissmann@rz.hs-fulda.de
†Department of Applied Computer Science
Fulda University of Applied Sciences, Fulda, Germany
Email: {sebastian.rieger, christoph.seifert, christian.pape}@cs.hs-fulda.de

***Abstract*—Over the last years, education paradigms developed from the traditional classroom learning to novel approaches like e-learning and blended learning. Especially blended learning, which combines the traditional approach with e-learning independent of time and place, is an important concept to increase the quality of study programmes. For the creation of laboratory setups in higher education dealing with computer networks, virtualized network environments have been continuously gaining momentum. Depending on the desired practical or theoretical orientation, they can be implemented using different paradigms, as well as corresponding hardware or software solutions. A high practical relevance and functional realism can be achieved using network emulation. However, emulation requires more resources compared to network simulators, due to the complexity of realistic network functions. To offer emulated virtual network environments, e.g., for a large number of participants in higher education courses, scalable virtualization backends and cluster solutions are necessary. In this article, we provide an overview over available paradigms to create networking experiments in higher education classes. We also present a number of requirements, which we identified to be important in the context of the networking laboratory (NetLab) of Fulda University of Applied Sciences. Based on these requirements, the available software solutions were compared and the best matching solutions were selected for the use in our laboratory. The scalability and performance evaluation of virtual network testbeds presented in this article, outlines the suitability of each compared network virtualization and emulation solution for higher education courses, and discusses possibilities for further improvements.**

Keywords–Network Virtualization; Network Emulation; Higher Education; VIRL; GNS3.

I. INTRODUCTION

In recent times, the paradigms for teaching in higher education have been evolving to include concepts like blended learning. This paradigm shift is especially helpful for hands-on laboratory exercises, as they typically include for example virtual testbed setups for larger class sizes as well as extending the use of the experiments and testbeds beyond the laboratory or classrooms. Furthermore, offering e-learning and blended learning content is a necessity today to increase the quality of study, e.g., by including complex and practical examples, and to keep pace with the rapid development of online courses and the mobility of students as well as lifelong learning. This is especially true for computer network labs, where real-world test setups are needed to complement lectures and theoretical models with practically relevant exercises [1].

To provide such environments, various approaches are possible, depending on the intended focus on theoretical or practical relevance. Figure 1 gives an overview of corresponding gradations of possible approaches, including references to some of the appropriate tools available. While the implementation of testbeds in real-world networks, e.g., campus networks of organizations and universities as shown at the left end of the figure, would provide the most realistic testbed, the risk of interference with regular operation and availability of the production network forbids this option in most cases. To overcome this problem, a separate physical testbed can be created apart from the production network. However, bootstrapping such a testbed with realistic characteristics is complex and expensive, hence making fast adaption to varying requirements and ever-changing network environments far from realistic. Virtual testbeds can reduce the setup cost immensely, but besides the additional effort and complexity introduced by the virtualization, still a lot of manual work is required for setting up and providing the required networking components and interfaces, virtual networks and so on.

Theoretical models, as shown on the right of Figure 1, take a completely different approach by abstracting complex network topologies and protocols in the model. This is specifically useful when designing experimental protocols, but the implementation of such formal specifications - or even the transfer of the insights learned - in the real world can be quite challenging due to missing practical orientation. Simulations, though also based on an abstract model of the network and protocols, improve the practical relevance by trying to replicate real-world characteristics. One of the big advantages of simulations is the capability of exact modeling of real-world behavior, such as timing or transmission quality. Another advantage of deterministic simulation is the option of changing the simulation speed. However, only an abstract network is modeled by a simulation, which does not fully reflect the characteristics of a real network.

To resolve these issues, emulation is recommended in [2] as the means of choice to implement virtual network testbeds. The authors come to this conclusion by evaluating the goals and advantages of emulation using the following criteria: *Functional Realism*, *Timing Realism*, *Traffic Realism*, *Topology Flexibility*, *Easy Replication*, and *Low Cost*. In their research, they show that emulation only lacks in the area of *Timing Realism* while fulfilling all other evaluation criteria. Therefore, emulation provides a flexible foundation for experimental network testbeds being positioned in the middle between practical implementations with high relevance for real-world environments and accurate but typically rather

abstract theoretical models. Besides the evaluation in [2], the advantages perceived by using emulation for virtual network testbeds with a high practical relevance is also described in [3], [4] and [5].

In addition, while lectures typically last about 90 minutes, the work in labs needs time for preparation and post-processing, which means that often only about 60 minutes are available for the hands-on exercises. This period is often not enough for in-depth exercises, especially when dealing with an abstraction or simplification of complex setups with high practical relevance. Also, the effort to provide testbeds increases significantly for large numbers of participants in laboratory courses. By choosing an emulation approach, much greater flexibility is gained also in terms of accessing the virtual test environment. Where formerly students had to attend in person, i.e., to be able to use the equipment on-site, they are now enabled to work basically from any location as long as sufficient Internet access is provided. Hence, emulated environments are currently primarily used for higher education courses in the networking laboratory (NetLab) at the Applied Computer Science department of Fulda University of Applied Sciences. These environments support the argument for emulation environments as presented in [2], which were also recently updated in [6]. The advantages of emulation-based environments, described in detail in the remaining sections of this article are also further amplified by the possibility to use them to improve the reproducibility of research experiments as described in [6] and also in [7] and [8]. This way, students in higher education, e.g., master courses, can get an insight into current research in the area of computer networks, use explorative learning to understand and discuss the findings, leading to a deeper understanding and developing the ability to carry out own small research projects and individually contribute to applied sciences, e.g., in papers adjacent to the master thesis. Emulated testbeds and experiments being developed by the students can in turn be used in research projects supporting the students to become junior researchers.

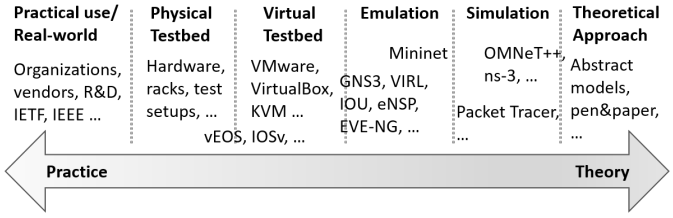


Figure 1. Classification of experimental scientific networking testbeds.

Nevertheless, the emulation of scalable virtual network testbeds results in high resource requirements due to the number of virtual machines needed, especially for large research projects or higher education courses. In this article, we present an analysis of these resource requirements of virtual network testbeds based on experiences from providing such an environment in the NetLab. Solutions that were evaluated and used in the NetLab are shown in Figure 1 (e.g., GNS3, EVE-NG, VIRL). Primarily, currently Cisco's Virtual Internet Routing Lab (VIRL) is used. Advantages of this solution will be presented in this article. Further, we discuss options to improve the scalability of such an environment.

The remaining part of this article is laid out as follows. The following Section II discusses related work in the area of virtualized networking laboratories and the effectiveness of such labs in higher education courses. Section III gives examples for emulated network topologies used in higher education courses in the NetLab. Based on these use-cases, requirements for virtual networking testbeds, and software products available to implement such testbeds, based on the classification and examples shown in Figure 1, are presented. Our experiences during the implementation of such a solution are discussed in Section IV, where we present our selection of tools for creating a scalable platform for virtual networking testbeds. Next, an evaluation of the scalability and performance of the VIRL environment is presented in Section V. As an alternative to VIRL, Section VI presents a comparison to the scalability and performance of GNS3 using the same virtualization environment and methodology as for the VIRL benchmark. Finally, a conclusion and future work can be found in Section VII.

II. Related Work

Services in today's cloud-driven infrastructures are based on sophisticated network topologies, which interconnect various network and server components, and build the foundation for scalability, redundancy and high availability of IT services. Given this practical relevance, it seems obvious that training in the lab is essential for computer science students to gain practically relevant knowledge of the theoretical concepts taught in lectures. Therefore, the simulation and emulation of such network topologies for teaching higher education classes as well as employing them in scientific projects is subject of current research. In [4], a virtual environment based on VIRL is compared to physical setups using Cisco Certified Network Associate (CCNA) pods and network simulation software like Cisco Packet Tracer. Furthermore, the cost, setup requirements and limitations are estimated and reviewed. The solutions discussed in this article address these limitations. A similar comparison is presented in [9], where the network emulator GNS3 is compared to Cisco Packet Tracer. The use of VIRL and GNS3 in the area of research and education are also discussed in [5] and [10], respectively. However, these papers do not address the scalability for VIRL and GNS3 in large higher education courses. Also, they do not discuss their didactical suitability and characteristics for the use in higher education. An extensible and scalable emulation/simulation framework based on a declarative XML-based language (as also used in VIRL) for modeling and evaluation of large network topologies is explained in [11]. This open-source toolset is also compared to other well-known simulation and emulation environments like ns-3 or PlanetLab. A brief introduction to model and simulate interconnected autonomous systems using the Boson Network Simulator is described in [12]. An open-source tool for simulating protocol behavior in congested networks by throttling network links and controlling delay and packet loss is described in [13]. This software is widespread and used in other emulation products to alter link transmission properties in network topologies. The didactics-oriented software called Netkit to model and emulate a wide-range of real-world devices is outlined in [3]. The goal of this emulation environment software is to setup networking experiments at low cost and with little effort. While these papers focus on large topologies and didactical suitability, they do not address the performance and scaling for large higher

education courses. A framework for reproducible container-based emulation of networking experiments is presented in [2]. In [14], network topologies are emulated by using virtual routers on Linux-based hosts. The real-time network emulator EmuNET used for testing protocol and application behavior in network topologies, is introduced in [15]. However, these solutions do not allow the use of real-world network operating systems and networks components (e.g., virtualized Cisco or Juniper equipment). The application of a cloud-based virtual environment for security related education is outlined in [16]. It is based on a platform called V-Lab, which utilizes open-source virtualization technologies, and software defined networking (SDN) solutions. Finally, an evaluation of the effectiveness of virtual laboratory environments for student learning is performed in [17]. In the paper, a course taught at the University of Massachusetts Amherst, which consisted of multiple lectures, homework assignments, and lab assignments, is evaluated. The paper tries to quantify student learning in lectures and labs, and the author concludes that learning indeed occurs almost equally as much during lab sessions as in lectures. This also coincides with the results of a study conducted by Stanford University researchers, who assigned the task of reproducing results from over 40 papers in the research area of computer networks to over 200 students [18]. They found that this kind of practical training is interesting for the students, and gives them the opportunity to understand topics of current research, or even interact with researchers. While these papers are useful for the evaluation of didactical suitability and characteristics of virtual laboratory network testbeds, they did not address the scalability and performance for the use in higher education courses discussed in this article.

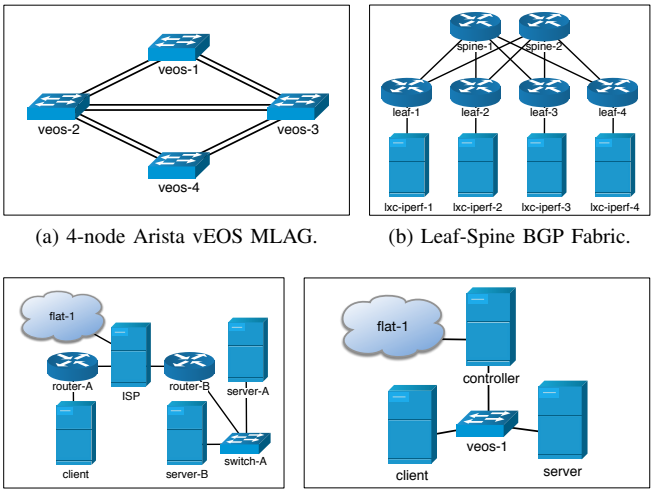
III. Virtual Environments for Network Testbeds

Following on from the consideration of emulation testbeds in [2], various possibilities of assisting lectures with practical exercises in our networking laboratory (NetLab), being described in this article, were evaluated. The following sections will first provide an overview of the approaches and the laboratory scenarios used in some of our courses, to present concepts and ideas behind the experiments carried out by the students. Afterwards, requirements for an implementation of a virtual environment are derived from the characteristics of these approaches and concepts. Finally, an overview of the candidates for a concrete implementation, currently used in the NetLab, are presented.

A. Examples for Exercises in the NetLab at Fulda University

The networking laboratory at the Applied Computer Science department of Fulda University of Applied Sciences provides practical relevant exercises in the field of computer network development, configuration and operation. Besides computer networks themselves, exercises also include distributed systems, e.g., internet, cloud or multimedia services, as well as network security or network management and monitoring. Thus, the laboratory supports lectures and further allows students to perform independent experiments to improve their knowledge and expertise in areas related to the indicated topics.

Refer to Figure 2 for some examples of network topologies used in our NetLab environment for student laboratory exercises. Figure 2a shows a topology consisting of four Arista vEOS (4.16.9) nodes with redundant links between them. The topology is one out of many used by master's students to understand and troubleshoot real-world networks. In this specific case, students team up to investigate the impact of the Spanning Tree Protocol (STP) in various data center networking scenarios using technologies like MSTP, LACP and MLAG. Based on similar exercises, the master's students also work on Leaf-Spine-based topologies including BGP fabrics, which are widely used in web-scale data centers by companies like Facebook or Microsoft (Figure 2b). Here, the endpoint nodes are based on Ubuntu 14.04 GNU/Linux containers (LXC), while the spine and leaf switches are driven by Cisco IOSv (15.6(2)T). In the past, we also implemented similar topologies using vEOS (BGP) and NX-OSv (FabricPath). A great advantage of this setup is the flexibility we provide for the students. LXC containers can be managed using standard Linux commands via SSH, nodes can be started and stopped in the middle of a running emulation, network links can be connected or disconnected, and it is also possible to configure various QoS parameters like delay, jitter or packet loss on the links. Beyond that, traffic entering or leaving a specific interface can be collected and investigated using Wireshark. SDN-based scenarios, including the OpenFlow controller *OpenDaylight* and OpenFlow 1.3 capable *Arista vEOS* switches, can be explored using the topology shown in Figure 2d. Since vEOS behaves identical compared to the EOS-based Arista hardware switches, by choosing VIRL over the previously used Mininet, students can test OpenFlow deployment in a near real-world environment.



(a) 4-node Arista vEOS MLAG. (b) Leaf-Spine BGP Fabric.

(c) Simple Toubleshooting. (d) vEOS/OpenDaylight SDN topology.

Figure 2. Examples of emulated network topologies for student exercises.

An example of a much simpler network topology is shown in Figure 2c. Students in bachelor programmes troubleshoot network misconfiguration (i.e., ARP, routing, delay, packet loss, port status) and discover the underlying network topology by using tools like ping, traceroute/mtr and Wireshark, before they establish connections to an Apache web server running on node server-B. Again, all server and client nodes are based on Ubuntu 14.04 LXC, while switches in this scenarios are based on IOSvL2 (15.2(4.0.55)E). For realistic WAN emulation we use the standard Linux network emulation *netem* on the ISP node to inject delay and add packet loss. The node named ISP

acts as a default gateway for the emulated topology, which is connected to the physical local area network for NetLab projects (flat-1). Thus, the invocation of commands like ping or traceroute targeting hosts on the Internet (i.e., google.com) is possible from inside the emulated environment, enhancing the practical relevance of the exercises. Furthermore, from within the NetLab, students can connect to their nodes in the emulated network using OpenVPN, for instance to configure the web server.

All examples and topologies mentioned in this section are under continuous development and are actively used in the NetLab. Corresponding topology configurations for VIRL are managed using Git and can be downloaded from [19].

B. Requirements for Virtual Networking Testbeds

The NetLab is currently equipped with 20 PCs and additional workspaces for notebooks, allowing students to bring their own devices to the laboratory to extend the lab equipment or to extend the experiments over the teaching time as described for e-learning and blended learning approaches in Section I. These workspaces are arranged in so-called ”islands”, so that students can work together in groups of four, and share a pre-packed experimental rack filled with networking equipment, such as routers, switches, and firewalls.

As mentioned in the introduction, working in the laboratory requires additional effort for preparation of the test setups (i.e., cabling the experimental racks and setting up an initial configuration) before the actual experiments can begin, as well as for cleaning up after the experiments are finished. Hence, often only 60 minutes are left for the hands-on exercises, which is not enough time for in-depth practical training, and the lab's state can not easily be preserved to span over multiple lessons when physical network components are being used. For this reason, in our networking lab we aim to provide a combination of classroom teaching and novel approaches like e-learning and blended learning to the students. Each of these approaches introduces demands on the learning environment, the time for preparation in the lab, or possibilities of external access.

- **Classroom teaching** — The traditional on-site learning in a classroom. In order to fulfill the time restrictions of classroom learning with a typical duration of 90 minutes for each session, the preparation of exercises to a predefined state must be considered. Further, the current state of an exercise should be storable and resumable at a later time to allow students to continue, discuss, share and present their work.
- **E-learning** — A teaching method that enables high flexibility in learning regardless of the times and locations of conventional classroom sessions. It must be possible to attend a course without ever visiting a traditional classroom. A fully virtual lab is required to allow students to login from home at any time and conduct an experiment using their own or individual resources.
- **Blended learning** — This can be seen as a combination of the previous paradigms, where students attend classroom sessions, but have the opportunity to finish exercises at any time by accessing the virtual lab from the NetLab outside of teaching time, but also from at home or any place providing sufficient Internet access. In addition, students are enabled to conduct their own experiments in order to deepen their teaching content.

The paradigm of blended learning offers continuing learning and collaboration between students together with lecturers and tutors, combining the advantages of classroom teaching and e-learning. Therefore, we followed this idea for the higher education computer network courses discussed in this article. We experienced an increase in student groups using the labs outside of regular class hours throughout the last semesters, supporting our decision.

Regardless of the paradigm chosen, the implementation must meet some of the requirements already outlined in [2], which are essential to ensure that the behavior of real networks and protocol peculiarities can be observed by students.

- **Functional realism** — Each system must provide the same functionality as its pendant in real hardware. Ideally, this can be achieved by executing exactly the same program code that is also used on real-world systems, i.e., by using real hardware or virtualizing the operating system images of real routers, switches and client systems.
- **Timing realism** — The timing behavior of all systems in the test bed should be indistinguishable from real physical systems. This is especially important for exercises where traffic behavior is inspected with tools like ping or tcpdump.
- **Traffic realism** — It should be possible to inspect real network traffic in the networking testbed. Client systems should be capable of generating and receiving network traffic from users and systems on the local network, or even on the Internet.
- **Topology flexibility** — It should be possible to create new topologies of various kinds without great effort, in order to be able to optimally map the requirements of the various courses.
- **Easy replication** — It should be possible to duplicate existing topologies without great effort, so that they can be used by different groups of students independently.
- **Scalability at low cost** — The environment must scale for large numbers of students, while at the same time minimize administrative effort and financial costs.
- **Didactical reduction** — The implementation approach must allow students to focus on the essential learning materials, as there is not enough time in classroom to understand complex simulation software before the actual experiment can begin.

Figure 3 shows the characteristics of different networking testbed approaches as introduced in Figure 1 and defined as well as evaluated in [2]. In the lower part of the figure, the didactical suitability of the approaches as perceived in several computer networking related courses in the NetLab was added. These didactical aspects are discussed in more detail in [20]. However, from this perception, it can be seen by evaluating positive (+) and negative (-) values while ignoring a neutral value (˜) that emulation is still the best option. Simulation (6 points) is offering better suitability for blended learning and e-learning, since it is lightweight, provides exact timing and the state can be prepared and restored even when using the

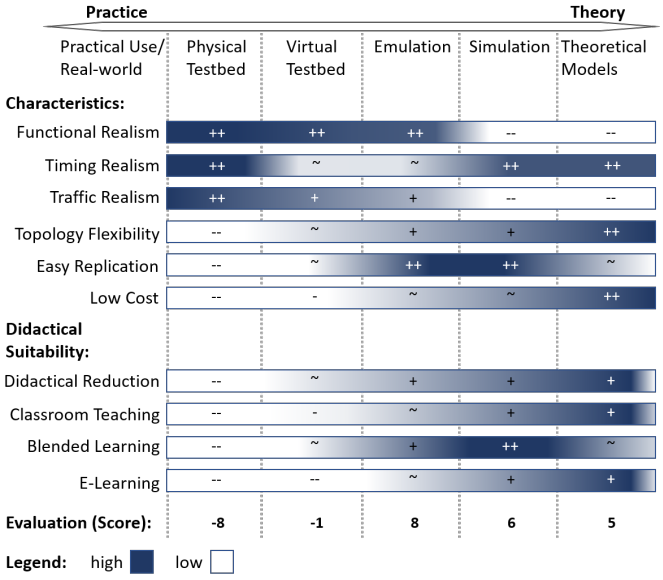


Figure 3. A comparison of networking testbed approaches.

course material remotely. This also holds advantages when using simulations in the classroom. Theoretical models (5 points), however, can be difficult to understand for students leading to a worse suitability for blended learning. As already discussed in this article, physical (-8 points) and virtual (-1 point) testbeds require much effort not only to build them, but also to prepare their use during lectures and laboratory courses. Hence, these options also received a low score in their didactical suitability observed in the NetLab. Regarding didactical reduction, emulation, simulation and theoretical models can use abstraction to hide the complexity, e.g., by using prepared experiments, models or formulas. Overall, emulation (8 points) still offers the best overall score, when taking the didactical suitability identified from courses in the NetLab into account.

C. Approaches for Implementing Virtual Network Testbeds

Blended learning techniques require ubiquitous access, as well as the possibility to easily comprehend and modify the experimental environment. Hence, integrated environments like simulators and emulators are the method of choice to meet the requirements of high realism and practical use.

Emulation can be seen as a compromise between theory and practice, especially when it is used to implement virtual network testbeds. It allows to deploy real-world operating systems (i.e., GNU/Linux servers, Windows clients) and utilize typical network management tools, like Wireshark or iperf. In the NetLab *physical and virtual testbeds*, as well as *emulation* and *simulation* have been used over the past years to support higher education courses and research projects. In accordance with the results discussed in [2], emulation has proven to be a particularly flexible solution. Physical testbeds are provided in the lab in form of pre-packed experimental racks to allow realistic student projects and Cisco certifications (i.e., CCNA, CCNP) [21]. However, due to the complex and time-intense preparation of the environment, these physical testbeds are not suitable for short-term exercises and lab sessions, in which students should carry out experiments, e.g., to see the practical use of theoretical concepts presented in a corresponding lecture.

Yet, the realization of *virtual testbeds* (i.e., using a distributed approach with VMware Workstation or a central approach with VMware vSphere ESXi) doesn't require less effort, as the preparation and maintenance of the virtual machines and networks is time-consuming. For this reason, in the NetLab *virtual testbeds* are mostly used for practically relevant client-server applications and experiments in the IT security area. The constant need to install software updates in the virtual machines used for these testbeds and adapt them to changes in the surrounding laboratory environment throughout the semester, requires additional effort. Simulation software like ns-3 [22] or OMNeT++ [23] is mentioned in some lectures in master's programmes, but not currently used for practically relevant experiments in the lab.

Practical training for the previously mentioned CCNA certification includes the use of Cisco Packet Tracer [24]. However, in some lecture exercises students criticized the missing practical relevance. For example, network clients (i.e., PCs) in Packet Tracer are simulated and do not provide feature-complete implementations of common network tools, such as arp, ping or traceroute. Another drawback of the simulation is that there are peculiarities, which only appear in the simulation and need to be specifically explained to students. One example of such behavior is that in case of an ICMP PING, the first packet will be dropped at the router in the destination network until the destinations MAC address has been determined using ARP. While this behavior is correct, it typically can't be observed in a real network, where almost any client starts to send packets to the router immediately after booting the operating system, hence its MAC address is already in the routers ARP table, when sending ICMP PING packets. Besides this lack in *Traffic Realism* and the stated lack in *Functional Realism*, e.g., due to missing common tools in the simulated clients and network components, there is also no *Timing Realism* achieved within Packet Tracer, which is an even bigger problem for research projects.

The software Mininet [25], mentioned in [26] and [2], is also provided in our NetLab, where it is mainly used for experiments in the SDN area. The resource requirements of Mininet are extremely low, due to a container-based approach, which allows to emulate huge topologies with hundreds or thousands of individual nodes. Therefore, Mininet is typically the best choice for large topologies and complex network management and automation implementations, e.g., in research projects. Still, the creation of complex topologies in Mininet requires decent knowledge of the underlying Python-API, which again is time-consuming in short-term courses. Further, it is not possible to use or connect arbitrary real-world network components in Mininet topologies, limiting the practical relevance of the experiments carried out in Mininet. Also, images of real-world network equipment, e.g., in form of virtual machines, cannot be used in Mininet. Hence, though the scalability and performance of Mininet is excellent, and its actively used in the NetLab for research projects and master's courses, it is not specifically considered in the evaluation of this article, due to the limitations for the use cases and virtual network topologies discussed in the previous sections.

IV. Rationale for Selecting VIRL and GNS3

Over the last years, Mininet [25], eNSP [27], GNS3 [28], EVE-NG [29] (formerly UNetLab) and VIRL [30] have been

deployed and evaluated as a solution for realistic emulation of networking environments in the NetLab. By the time of publishing our initial research [1][20], VIRL was the most promising option, and was already used in several courses. It was compared to other emulation software alternatives based on criteria that are directly derived from administrative and educational requirements presented in Section III-B. In the meantime, new major versions of GNS3 (v.2.1) have been released, which implement features like QoS properties that we missed so far. Figure 4 depicts an updated revision of the initial comparison table, which is simplified compared to our initial paper and now includes the new version of GNS3. It clearly shows VIRL as the most promising approach when compared to GNS3 (v.2.0). However, with the new software release, GNS3 (v.2.1) even gets a slightly higher score.

Cisco Modeling Lab (CML), which is able to scale for a large number of nodes providing centralized management and automatic load balancing, is mainly depreciated due to the high licensing costs. CML is using the same technical basis as VIRL, and can be considered as the multiuser version of VIRL. In the case of EVE-NG, we were only able to test the free version, which lacks in some required functionality, such as centralized management, automatic load-balancing, but more importantly, the possibility to connect to a physical network, and to allow incoming VPN connections. These functions, however, might be available in the upcoming Premium or Learning Center release, which was announced to be released in 2018.

The poor performance of Mininet in our comparison is mainly due to the fact that no custom images are supported and connections to the console are very limited, hence the demands for functional realism and ubiquitous learning are not fulfilled for the use cases described in Section III-A. In addition, due to the implementation of Mininet as a command line tool and the missing GUI, the collaboration of students as well as the required training time in the laboratory is worse compared to the considered alternatives GNS3 and VIRL. However, even though the application of Mininet is not suitable for our virtual lab, we provide it locally installed on the lab computers, where it is especially used for thesis work in the field of SDN and NFV as well as for master's courses. By using LXC containers and OpenVSwitch as the basis for virtual hosts and networks, Mininet allows large network topologies to be launched on single-user computers even with limited resources. Due to its lightweight approach and its excellent suitability as an SDN environment, Mininet remains the best choice for research-oriented experiments in the master's field despite its lower rating compared to GNS3 and VIRL.

The advantages of VIRL and GNS3 are strongly related to the findings in [2] and [5]. The functional realism of VIRL is extended compared to the other alternatives, as it allows to use officially licensed network operating system images of Cisco components, like IOS or NX-OS, while at the same time, images from other vendors (i.e., Arista vEOS, HP VSR, Juniper vSRX/vMX, Cumulus VX) are supported as well. However, the most important advantage over the other alternatives is the underlying scale-out architecture based on OpenStack. This allows a central and scalable installation and enables the users to access the emulated network topologies location-independently (i.e., from within the NetLab or from at home using private PCs and laptops. Multiple VIRL hosts in the NetLab enable a scale-out of the testbed, which allows to emulate much bigger topologies than possible on a single PC in the laboratory or on a student notebook. In addition, the open architecture of OpenStack, as well as the open components used by VIRL (i.e., Ubuntu 14.04, LXC, linux-bridge, VXLAN) make it possible to extend the environment with in-house developed components to build specifically tailored testbeds for the use in research and education. Still, for the operation of VIRL in our environment, a few extensions were implemented, including customizations to the Arista vEOS and CumulusVX operating system images for our university's environment and for deployment in VIRL [31]. For example, to allow the operation of MLAG, it was necessary to modify the *base_mac* in order to prevent clashes of MAC addresses generated by vEOS with locally generated KVM MAC addresses [31].

In the latest version, GNS3 offers a lot of the functions required in our lab environment. Regarding network operating system choices it is as flexible as VIRL, with most vendors providing software images that are free to use in GNS3. However, especially Cisco requires users to buy a regular software maintenance contract to legally use their software images in this virtual environment. Beyond network operating system images, GNS3 supports the use of a wide range of server, desktop, and mobile operating systems by QEMU virtual machines. Although, cluster support with automatic scheduling of nodes in a topology (or project in GNS3) across multiple compute nodes is still not supported, the abandoning of Ciscos academic license for VIRL raised the importance to consider GNS3 as an alternative for our virtual network testbeds.

Improvements in terms of scalability (i.e., the size and amount of emulated testbeds), performance (i.e., the time to bootstrap a complete emulated topology) and usability (i.e., initial configuration) of VIRL and GNS3 will be discussed in the following sections.

V. Performance and Scalability Evaluation

Thanks to using virtual networks for the exercises shown in Figure 2, students can especially benefit from the advantages of the emulation as discussed in Section I. However, for complex topologies and a large number of students in the class, the benefits of the emulation places enormous demands on the virtual infrastructure it is running on. Even for smaller topologies it can take more than 5 minutes to start the emulations. Therefore, in the following sections, we describe a way to benchmark and optimize the waiting time until the emulations are ready to be used by the students in our laboratory.

A. Implementation of a VIRL Benchmarking Environment

The hardware we use for evaluating the performance and scalability of our VIRL environment was described in detail in [32]. For the evaluation presented in this article, initially VIRL 1.2.83 (October 2016) was used, since we kindly received an extended node count license in the Cisco dev/innovate research program for this version. Later, we repeated the evaluation with VIRL version 1.3.296 (August 2017) leading to slightly better, but similar results, as shown in the following sections of this article. All VIRL hosts are based on Ubuntu 14.04 VMs, each configured with 32 vCPUs and 64 GB of RAM. These VMs build a nested virtualization environment inside a

		Mininet		VIRL		CML		GNS 3 (v.2.0)		GNS 3 (v.2.1)		EVE-NG	
Criteria	**Weight 1-3**	**Description**	**#**	**Description**	**#**	**Description**	**#**	**Description**	**#**	**Description**	**#**	**Description**	**#**
Administrative Effort													
- License (Cost)	3	None (Open Source)	1	Cheap (199 € per 20 Cisco nodes per year)	0	Expensive (> $ 2.500 per 25 nodes per year)	-1	None (Open Source)	1	None (Open Source)	1	Free (Premium/Learning Center Edition?)	1
- Centralized Management	2	Isolated installation (Scripting support)	0	Central managed cluster with multiple compute nodes	1	Central managed cluster with multiple compute nodes	1	Multiple isolated servers	0	Multiple isolated servers	0	Perhaps with Premium/Learning Center Edition	0
- Compatibility and Accessibility	2	GNU/Linux (Virtual machine for Windows and MacOS)	0	VMMaestro-Client, and VMs for GNU/Linux, Windows, MacOS, vSphere	1	VMMaestro-Client, and VMs for GNU/Linux, Windows, MacOS, vSphere	1	GNU/Linux, Windows, MacOS	1	GNU/Linux, Windows, MacOS	1	Web-UI	1
- Custom Images	2	No support for real router and switch firmware images	-1	Extendable (node restriction only for included Cisco nodes), third-party images importable	1	Extendable (node restriction only for included Cisco nodes), third-party images importable	1	Images for routers and switches needed (Cisco Images not included)	1	Images for routers and switches needed (Cisco Images not included)	1	Images for routers and switches needed (Cisco Images not included)	1
- Load Balancing	2	Manually	0	Yes	1	Yes	1	Manually	0	Manually	0	Perhaps with Premium/Learning Center Edition	0
- Required Compute Resources	1	Low (LXC containers)	1	High (device dependent)	0	High (device dependent)	0	Medium (device dependent)	1	Medium (device dependent)	1	Medium (device dependent)	1
Educational Requirements													
- Connect to Physical Network	1	Yes	1	Yes	1	Yes	1	Yes (NAT with GUI, Bridged requires CLI configuration)	1	Yes (NAT with GUI, Bridged requires CLI configuration)	1	Only manually on CLI	0
- Multiple Users and Sessions	2	Manually	0	Yes	1	Yes	1	Single project per user, multiple user sessions from different clients possible	0	Single project per user, multiple user sessions from different clients possible	0	Yes, single project per user, multiple user sessions perhaps with Premium Edition	0
- Console Session	1	Limited (LXC console)	-1	Yes (singleuser)	0	Yes (singleuser)	0	Yes (multiuser support)	1	Yes (multiuser support)	1	Yes (multiuser support, but topology only shown in one browser)	1
- QoS properties for links	2	Yes	1	Yes (Only delay and packet loss)	0	Yes (Only delay and packet loss)	0	Only manually on CLI	0	Yes	1	Only manually on CLI	0
- VPN Connection to Virtual Networks	2	Manual configuration	0	Yes	1	Yes	1	Yes, with VPN server in the virtual network	1	Yes, with VPN server in the virtual network	1	No	-1
Weighted Score			**4**		**13**		**10**		**12**		**14**		**7**

Figure 4. A comparison of network emulation software based on out requirements.

VMware vSphere 6.5 cluster, in which each of the four VMs is bound to a separate physical ESXi host by DRS constraints to limit fluctuations in available resources in the virtualization environment. Each underlying ESXi host is equipped with two 8-core Intel(R) Xeon(R) E5-2650v2 2.60 GHz CPUs, 256 GB RAM and uses two NetApp E2700 over a redundant 16 Gbit/s Fibre Channel connection as a storage back end. The nodes are connected via 1 Gbit/s Ethernet to a Cisco Catalyst 3850 switch and with two 10 Gbit/s links to an Arista 7150S-24 and Arista 7050S-52 leveraging MLAG. As discussed in [2], the *Timing Realism* regarding emulation with regard to virtualization particularly depends on the isolation level of the virtualization environment. When strict isolation is not guaranteed, concurrently running VMs can have a negative performance impact. Therefore, we defined a separate resource pool with static resource allocation for our VIRL environment. All benchmark scenarios were repeatedly performed at night in the semester break to ensure minimum load and interference of the workload on the ESXi cluster. By monitoring the overall performance and capacity of our VMware vSphere cluster, we were able to verify that VMs related to the VIRL benchmark were the only systems that produced considerable load in our VMware environment during the tests. The topology shown in Figure 2a was used to evaluate the performance and scalability of our environment. Due to its small size and node count, it can be scaled fine-grained up to the full capacity of our cluster. For the evaluation of the scalability of virtual testbeds and their application in higher education courses with a large number of participants, the following four metrics are of special interest:

- *Start Time*, the time until the start of all submitted topologies is processed by VIRL's REST API
- *Active Time*, the time until all VMs start to boot
- *Usable Time*, the time until booting has finished and the virtual console of all vEOS nodes is accessible
- *Console delay*, the latency of the virtual console

To measure these metrics and in order to minimize outliers, we developed a script to run our performance tests in a reliable and reproducible manner. Each test run first starts up the network topologies using VIRL's REST API and measures the time until the start is confirmed (*Start Time*) and all nodes become active (*Active Time*). In our terminology, *Active Time* means that the VM is deployed by the OpenStack Nova scheduler on a VIRL host system, all virtual networks and ports are up and accessible, the vEOS image is provided and its boot sequence starts. Next, we measure the time until the VM really becomes responsive by connecting to the virtual console (*Usable Time*) and finally measure the interaction delay of keyboard inputs (*Console delay*). For this purpose, a Python script was developed, which establishes WebSocket connections to the serial consoles of the nodes running on the VIRL hosts.

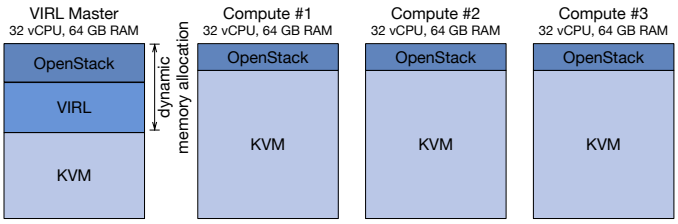


Figure 5. Schematic view of the environment's memory usage.

A schematic representation of the VIRL environment is depicted in Figure 5. At first, we performed all tests on only a single VIRL node, meaning that the node not only executes the

VMs needed for the emulated topology, but also acts as control node, which provides the OpenStack and VIRL environment. In Section V-B, we will share some negative observations we made, when including the control node in VM execution. Each individual test run for an increasing number of simultaneously emulated network topologies was executed ten times in a row. We started with only one topology and scaled in steps of five, until the VIRL host was working to capacity. Next, all tests were repeated on a 2-node and finally on a 4-node VIRL cluster to draw conclusions concerning scalability of the environment. The benchmark script and related toolset is available at [33].

B. Scalability Evaluation

The results from our previously explained test case are illustrated in Table I and Figure 6. When using a single VIRL host, the maximum number of simultaneously emulated network topologies is mainly limited by the resources (primarily the amount of main memory) available to the host. Each vEOS node uses 1 vCPU and 2 GB of RAM. Therefore, a test run with ten concurrently emulated network topologies requires 80 GB of main memory (10 * 4 vEOS nodes * 2 GB RAM) to be available, which is more than a single VIRL host in our test system can provide (see Figure 5). Hence, a stable execution was not possible with a single host, even with memory overprovisioning enabled (Figure 6a).

TABLE I. RELATIVE CHANGE IN *Start Time*, *Active Time* AND *Usable Time* DEPENDENT ON THE NUMBER OF TOPOLOGIES.

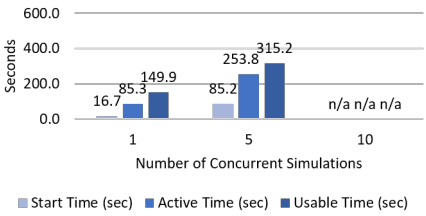
Number of Topologies			*Start Time*	*Active Time*	*Usable Time*
1	→	5	5.0617	2.5242	1.7440
5	→	10	2.0378	1.8517	1.6705
10	→	15	1.5105	1.4693	1.4686
15	→	20	1.3597	1.4548	1.4420
20	→	25	1.2334	1.2797	1.2793
25	→	30	1.2081	1.2304	1.2349

Looking at the effects of the number of parallel topologies in respect of performance, Figure 6b clearly depicts our expectation of a linear increase of the *Start Time*, while with an increasing number of topologies the *Active Time* and *Usable Time* ascends non-linear. The effect can best be observed when performing the test on a cluster with four or more nodes (Figure 6c). Up to a number of 10 simultaneously started topologies, the difference between *Active Time* and *Usable Time* gets smaller. This can be explained by the overhead the OpenStack-based resource scheduling and management introduces, which decreases with the number of simultaneously started topologies. For higher numbers of concurrent simulations, the system load generated from setting up virtual networks and interfaces causes an increased difference between *Active Time* and *Usable Time*. The curve for *Usable Time* has a comparatively steep slope as expected, as for an increasing number of virtual nodes, the time until all nodes are usable increases due to the limited resources.

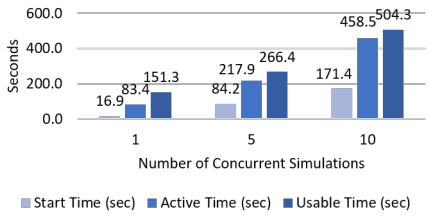
Alongside with the overhead introduced by the scheduling, the comparatively large difference between *Start Time* and *Active Time* results from the expensive process of creating all required virtual networks for connecting the devices. All links of the topology (Figure 2a) are redundant, which requires the creation of two VXLAN segments and its associated ports per pair of devices on the GNU/Linux bridge interface of the OpenStack nodes. The overhead of this was clearly visible by the CPU load produced by the Neutron process on the OpenStack controller node. The main limitation here is the fact that neutron-server and nova-conductor are single-threaded in VIRL's OpenStack Kilo setup, which limits the maximum performance of virtual network creation to a single CPU core. In most of our test cases, the CPU core neutron-server was executed on, was working to capacity. To overcome this limitation, we increased the number of neutron-server, nova-api and nova-conductor worker processes to ten. However, due to the fact that memory gets reserved on the VIRL master for these processes, the additional resource requirements resulted in a decrease of the maximum number of simultaneously started topologies to only 25 (Figure 5). Even more problematic was the observation that some of the vEOS nodes were not successfully started at the end of the test run, which is most likely explained by the dynamic memory allocation. When starting all topologies nearly at the same time, nova-scheduler is not able to determine the truly remaining amount of main memory and schedules too many VMs to the control node. At the same time, Figure 6d depicts that the *Usable Time* of the 20 topologies decreased by 16% and *Active Time* by 18%. While we assume room for improvement by carefully optimizing the OpenStack and KVM configuration, our recommendation is rather the use of a dedicated VIRL master node, which is currently not possible in VIRL, but can be manually achieved by deactivating nova-compute on the controller. The average console delay of the emulated vEOS nodes stays nearly constant with a growing number of simultaneously active topologies, as shown in Figure 6g. As a result, even if the time to start the concurrent simulations increases, a smooth use of the individually usable emulations is guaranteed despite the increased CPU load of the hypervisors. Limiting factors regarding our benchmark are more related to the amount of main memory and I/O performance, rather than the CPU load. What accounts for the latter is primarily the OpenStack and VIRL management processes, as well as the boot process of the vEOS instances, which utilizes the assigned vCPU to its maximum capacity for about 60 seconds in case of our test setup. As a performance improvement, Cisco recommends the use of a ramdisk for running Nova VMs, as well as an SSD for the VIRL hosts. We implemented both recommendations in our test environment to compare the impact on performance. First, a ramdisk was created, which is regularly only supported on the controller in VIRL, hence we needed to manually configure it also on the compute nodes. Figure 6e depicts the measurement results, clearly showing only a minor performance improvement, which is obviously attributable to the small amount of required ephemeral storage of only about 213 MB for a vEOS image. Second, we added two local solid state drives (Samsung 850 PRO) to each of the servers. Figure 6f shows no significant improvement of the performance, which is attributable to the previously used storage back end (NetApp E2700) already offering about 650 MB/s read/write performance. Due to the higher number of IOPS of the SSD, we assume that an improvement is likely to be observable when the I/O load of the VMs increases as a result of more complex topologies.

VI. EVALUATION OF GNS3 AS AN ALTERNATIVE

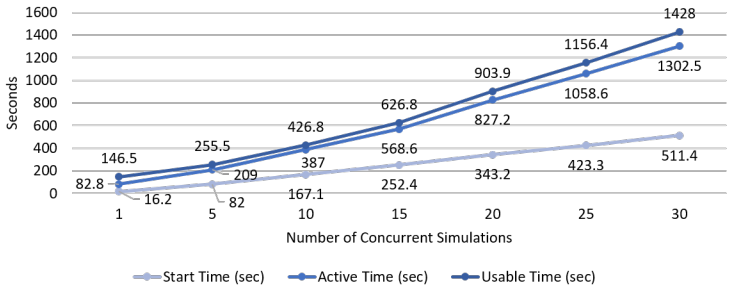
During winter term, we ported the example topologies for the advanced computer networks masters' course, as introduced in Section III-A, to a GNS3 testbed. We experimented with GNS3 v.2.1.3. The production environment described in



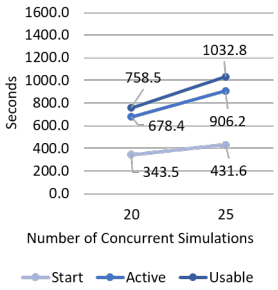
(a) Single VIRL Host.



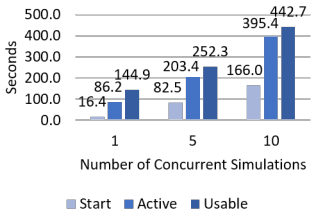
(b) 2-node Cluster.



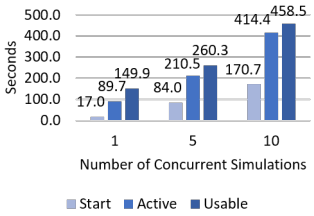
(c) 4-node Cluster.



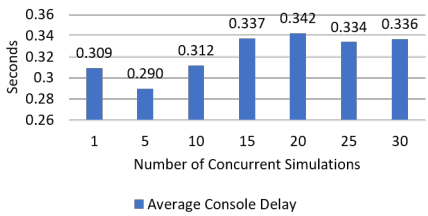
(d) 4-node, More Workers.



(e) 4-node, Ramdisk.

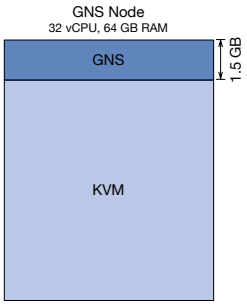


(f) 4-node, SSD.

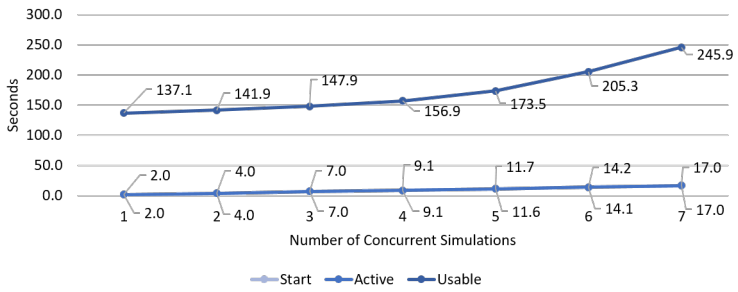


(g) 4-node, Avg Delay.

Figure 6. Results from measuring the time it takes to start multiple instances of the topology shown in Figure 2a.



(a) RAM usage.



(b) Measured time to start multiple instances of the topology from Figure 2a in GNS3.

Figure 7. Evaluation of GNS3.

this section, however, currently still uses v.2.0.3. The GNS3 testbed consists of five VMs, using an identical configuration as described for the VIRL testbed in Section V-A. Due to the missing automatic clustering and scheduling of started topologies in GNS3, we only used one GNS3 VM as a single node for the initial evaluation of the GNS3 performance described in this section. We started the same diamond-shaped topology with four Arista vEOS nodes, shown in Figure 2a, as for the VIRL evaluation discussed in the previous section. The entire process to start the topologies and the handling of so-called projects in GNS3 is different, compared to VIRL. Also, not only the code quality (e.g., regarding documentation and comments as well as the stability of the GNS3 GUI) still seems to be significantly lower on the GNS3 side. However, the progress of the project is impressing and not only the unattractive license model of VIRL for the use in academia strengthens the potential of GNS3 compared to VIRL. Since the code for GNS3 is available as open-source (under GPL-3.0 license) in a GitHub repository, extensions as well as fixes to the environment are possible. Although, GNS3 does not seem to focus classroom environments, as described in the requirements for a cluster environment in Section III-B. Instead, GNS3 seems to focus on a single user environment (e.g., for network consultants). However, features like the

simultaneous shared access to the console of emulated devices by multiple students, hold advantages and unique functions offered by GNS3 compared to VIRL when being used in classroom environments.

To get reproducible evaluation results for the performance of GNS3 in the same way and using the same requirements as described for VIRL in Section V-B, a Python-based benchmark solution was implemented to automatically run the experiment. Again, benchmarks were run ten times and GNS3 results in Figure 7 and Table II show the average time of these runs. Furthermore, benchmarks were run in different timeframes, to reduce possible side effects of the underlying virtualization infrastructure, which was not experiencing any other workload peaks during the tests. As for VIRL, the source of the benchmark process can be downloaded from our Git repository [34]. The Python script uses the GNS3 REST API [35] to create benchmark projects (as copies of the original topology) and measuring the time for this process (analog to the *Start Time* described in this article for VIRL). Afterwards, the benchmark projects are started and the time until all nodes in the topologies are active is measured (as for *Active Time* described in previous sections of this article). Since GNS3 is not dependent on another scheduling solution, as for example OpenStack in the VIRL setup, *Active Time* was reached only a fraction of a second after *Start Time*. This is due to the fact that GNS3 directly created the QEMU/KVM processes after topologies were started without the overhead of scheduling and messaging between the OpenStack components as described in Section V-B.

Also, as shown in Figure 7b *Usable Time* was reached quicker compared to VIRL on a single node. This is influenced by the smaller CPU and RAM footprint of GNS3, shown in Figure 7a compared to VIRL as depicted in Figure 5. This smaller resource footprint and corresponding shorter duration of benchmark runs, allowed for more fine-grained steps of scaling the number of concurrent simulations. While *Start* and *Active Time* increased linearly, as already described in this section, similar to VIRL, *Usable Time* increased rapidly, as soon as half (4 concurrent simulations * 4 vEOS nodes * 2 GB RAM = 32 GB RAM) of the memory was filled by the vEOS VMs. As expected, we were not able to run a significant amount of benchmarks with 8 concurrent simulations (filling up the entire 64 GB RAM), although Kernel Samepage Merging (KSM) was used to compress identical memory pages of the VMs. Since KSM's memory deduplication also takes resources and especially time to scan allocated pages, only a few runs were successful using 8 concurrent simulations. Their *Start Time* varied from 13 to 20 seconds, *Active Time* was between 14 and 20 seconds, *Usable Time* between 291 and 331 seconds, fitting into the trend of Figure 7b. The relative change in *Start Time*, *Active Time* and *Usable Time* for increasing numbers of concurrent simulations is listed in Table II. The last line of Table II can be indirectly compared to the same increase in concurrent simulations for VIRL from Table I, though the table for VIRL is based on a setup with 4 nodes while the GNS3 benchmarks were run on a single host. For this reason, the comparably large increase of *Start Time* and *Active Time* for GNS3 is plausible. However, the lightweight implementation and smaller resource footprint of GNS3 results in a lower increase of *Usable Time* even when GNS3 running on a single node is compared to a 4-node VIRL cluster.

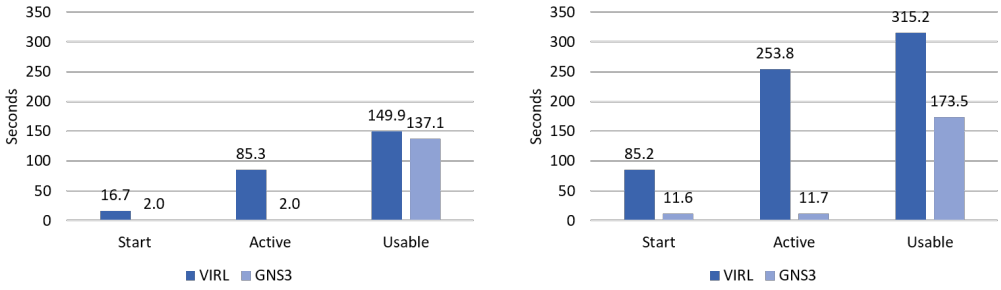
TABLE II. RELATIVE CHANGE IN *Start Time*, *Active Time* AND *Usable Time* DEPENDENT ON THE NUMBER OF TOPOLOGIES USING GNS3.

Number of Topologies			*Start Time*	*Active Time*	*Usable Time*
1	→	2	2.0000	2.0000	1.0350
2	→	3	1.7500	1.7500	1.0423
3	→	4	1.3000	1.3000	1.0609
4	→	5	1.2747	1.2857	1.1058
5	→	6	1.2155	1.2137	1.1833
6	→	7	1.2057	1.1972	1.1978
1	→	5	7.3000	7.3000	1.2416

Figure 8 compares the results of one (Figure 8a) as well as five (Figure 8b) concurrently started topologies between GNS3 and VIRL. To allow a comparison, the results shown in the figures are taken from a single VIRL node, which was shown in Figure 6a in Section V-B, so that GNS3 and VIRL both used only one node with an identical setup and the same environment as described in Section V-A. It can be seen from the figures that not only the *Start Time* and *Active Time* is smaller for GNS3, due to the fact that GNS3 starts all QEMU/KVM processes for the vEOS nodes directly, but also and more importantly, the *Usable Time* decreases significantly especially for large numbers of concurrent simulations. For example, as shown in Figure 8b, the same topology that was started five times in VIRL was usable after 315.2 seconds, while being available for the students after 173.5 seconds (55% of the time taken in the VIRL setup) when using GNS3 running in the same virtualization environment with the same dedicated resources.

Given the smaller resource requirements of GNS3, we tried to identity the limiting factors for the results we measured in further experiments, where CPU and RAM resources were systematically reduced. Figure 9 shows CPU and RAM usage of the GNS3 VM. Additionally, in the middle of the figure, the CPU load on the underlying host is depicted to check that there were no significant additional workloads in the host while running the benchmarks. In the timeframe from 07.01.18 10:40 to about 13:45, a benchmark using the standard value of 32 vCPUs and 64 GB, as for the VIRL tests, was used. It can be seen in the figure that the VM experienced high CPU ready time and less active time. Investigating this effect further, led to the finding that this impact was caused by a hyperthreading overhead combined with the NUMA architecture of the underlying host (2 CPU sockets with 8 physical cores each). This means that during the start of a large number of concurrent simulations, hyperthreaded cores competed against each other and also led to the effect that they needed to access RAM on another NUMA node, further increasing ready time and access delay. We were able to mitigate this effect, by decreasing the virtual CPUs available to the VM from 32 to 16, so that the cores used by the VM could be placed mostly on the same NUMA node. The result can be seen in Figure 9. Two benchmark runs, one from ca. 13:45 to 19:45 and one from ca. 19:45 to 1:45 are graphed in the figure.

The results measured for these runs are depicted in Figure 10. As visible, the results are similar to the runs with 32 vCPUs, shown in Figure 7b. However, as shown in the graph for the CPU load on the VM in Figure 9, the CPU of the underlying host is nearly used up to its capacity for these runs, with the ready time of the CPU being reduced to a value close to zero. Hence, the overall load of the benchmark on the underlying virtualization infrastructure was less, though



(a) 1 Simulation. (b) 5 Concurrent Simulations.

Figure 8. Comparison of starting the topology shown in Figure 2a on a single VIRL and GNS3 node.

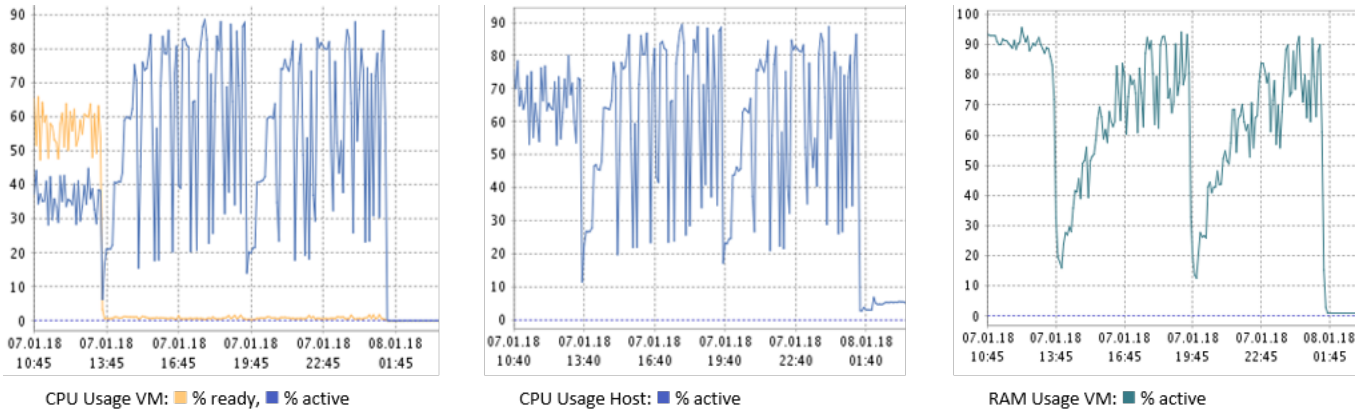


Figure 9. CPU and RAM usage of the GNS3 VM and the underlying host during benchmarks shown in Figure 7b and 10.

the results for our measurements did not change significantly. From the two full benchmark runs shown in Figure 9, the CPU and RAM resources consumed by the tests are also visible. The benchmark run started around 13:45 first shows steps that result from first 10 times repeatedly starting a single topology, then 10 times 2, followed by 10 times 3, and afterwards 10 times 4 concurrent simulations. As stated before, a single topology uses 4 x 1 = 4 vCPUs and 4 x 2 GB = 8 GB of RAM. Hence, 4 concurrent simulations used 16 vCPUs and 32 GB of RAM. Therefore, vCPUs were the most significant limiting factor for our benchmarks, even after the issue described for concurrent hyperthreaded vCPUs across NUMA nodes was resolved. As shown runs with more than 4 concurrent simulations, starting at around 15:15 (1.5 hours after the benchmark was started), account for ca. 75% of the entire 6 hours that the benchmark took to complete. In the right part of Figure 9, the RAM consumed by the VM is depicted. Along with the CPU usage, also the RAM consumption increases with the amount of concurrently run simulations, as expected. It can be seen in the picture that the RAM of the VM is slowly starting to saturate, after the maximum capacity of concurrent simulation is reached.

Booting a single vEOS 4.17.5M instance in our environment already takes ca. 135 seconds until the prompt is usable. Therefore, the results measured for the start of a single instance of our topology with four vEOS nodes is still close to the minimum time that a single vEOS switch needs to boot. As discussed, the amount of vCPUs is the main limitation in our scenario. To further investigate the influence of the vCPUs on the start of the concurrent topologies, we reduced the number of vCPUs available to the GNS3 VM further down from 16 to 8. Results after this degradation are shown in Figure 11. As expected, the further reduction of vCPUs moves the point where *Usable Time* starts to rise, increasing the slope of the curve, from 4 to 2 concurrent simulations, as 2 x 4 x 1 = 8 vCPUs of the vEOS instances fill up all virtual CPU cores available for the VM.

To crosscheck a possible influence of the RAM limits of the VM, additionally, the experiment was again repeated 10 times in different timeframes using only 32 instead of 64 GB RAM for the VM while changing the vCPUs back to 16. Results of this experiment are shown in Figure 12. By comparing Figure 10 and Figure 12, it can be seen, as expected, that neither *Start*, *Active* nor *Usable Time* were significantly impacted by limiting the RAM to 32 GB, as long as not more than 3 concurrent simulations were started. Starting 4 concurrent simulations already experienced a slightly higher *Usable Time*. Benchmarks with more than 4 concurrent simulations were, as expected, not possible, due to the fact that 4 concurrent simulations already consume 4 x 4 x 2 GB = 32 GB RAM.

Evaluating the comparison running the same benchmark process for GNS3 as for VIRL, GNS3 not only has the advantage of being open-source and hence highly customizable as well as offering an attractive licensing model compared to VIRL and Cisco Modeling Lab (CML) for the use in higher education, it also offers better scalability due to its significantly smaller resource footprint. However, the lack of a cluster solution as well as classroom features (e.g., user management) for GNS3 is still leaving some advantages on VIRL's side. Nonetheless, manually distributing nodes of a single topology

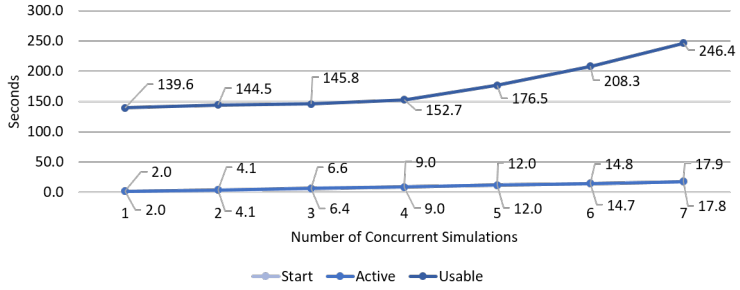


Figure 10. Results with 16 instead of 32 VCPUs for the GNS3 VM.

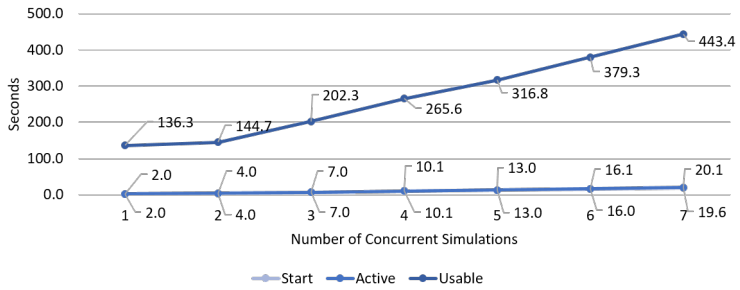


Figure 11. Results with 8 instead of 32 VCPUs for the GNS3 VM.

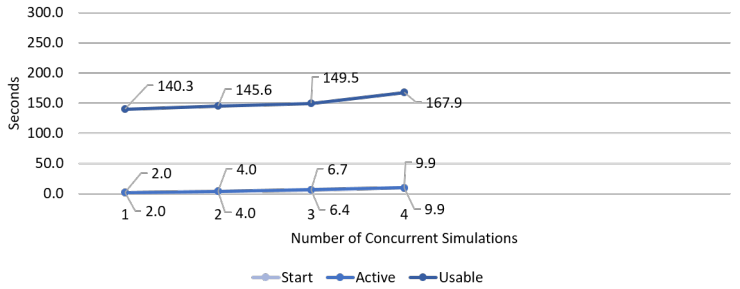


Figure 12. Results with 16 instead of 32 VCPUs and 32 instead of 64 GB RAM for the GNS3 VM.

run in GNS3 across multiple servers is possible. While not being able to dynamically spread the load evenly across all nodes, this already allows for fine grained scaling of GNS3 environments as back ends for higher education networking courses. Also, clustering and collaboration support seems to be on the road map for future GNS3 releases. Comparing GNS3 and VIRL regarding their features, as for example given in [20], recent versions of GNS3 already introduced previously missing features for the use in education. For example, since version 2.1.0 controlling the delay, jitter and packet loss of links, e.g., for WAN emulation, which was previously already available in VIRL, is now also possible in GNS3 projects. Version 2.1.1 introduced the display of the server individual nodes of the topology are placed on. Further tracking of the RAM and CPU usage of the started topology in GNS3 is planned for the upcoming GNS3 web interface [36]. Overall, despite some glitches and the difference in code and documentation quality, GNS3 looks like a promising alternative for future virtual testbeds for computer networks in our NetLab.

VII. Conclusion and Future Work

Cisco VIRL provides a platform, which is capable of creating realistic and scalable virtual network testbeds for education and research projects. In comparison with alternatives, such as GNS3 or EVE-NG, a clear advantage is that it offers to use original Cisco operating system images in conformance with license requirements. Beyond that, an even more important feature is the foundation of VIRL, which is based on the open-source project OpenStack. This enabled us to modify and extend the environment as shown in this article, and to build a well-scaling multi-node VIRL cluster, which supports a sufficiently large number of simultaneous emulations for application in education. Further, by allowing the utilization of standard network management applications (i.e., ping, traceroute) and operating systems (i.e., Ubuntu VMs) inside the emulated network testbeds, as well as the connection to real physical networks, a great flexibility and functional realism can be achieved in comparison with other simulation approaches. The increased start time introduced by the emulation, especially for complex topologies, can be compensated by using a VIRL scheduler that we developed to specifically address the requirements of our NetLab [37]. It offers to pre-load topologies based on a schedule, e.g., in advance of an upcoming seminar, hence minimizing delays for the students. Additionally, we are actively developing a management application for VIRL and GNS3 labs. When finished, it will provide a self-service system enabling students

to subscribe to courses and to start working on topologies and reserving virtual lab time. To increase performance even further, the most promising approaches are given by increasing the number of cluster nodes and optimizing to the OpenStack resource and network management.

By the time of writing, new major versions of GNS3 were released. Still, GNS3 does not provide a way to use Cisco operating system images in conformance with the license requirements. However, a large number of third-party virtual network equipment images is available. Sadly, a load balancing of started projects across multiple hosts still is not possible. In this article, we present a comparison of the scalability and performance of GNS3 as an alternative to VIRL. Though the overall quality of GNS3 seems to be lower compared to VIRL, performance and features in recent versions have surpassed the performance as well as educational suitability of VIRL for the use cases described in this article. Since Cisco seems to have changed the strategy for VIRL and tries to move universities to the expensive Cisco Modeling Lab, the new version of GNS3 could become an interesting alternate candidate for our environment. We are currently looking into the possibility to develop appropriate extensions to GNS3 or EVE-NG to fix the current lack in cluster-based load balancing and centralized management and real-time collaboration options on running simulations compared to VIRL.

ACKNOWLEDGMENT

We thank Cisco for providing us with a research license within the context of the Cisco dev/innovate research program for our Virtual Internet Routing Lab (VIRL) cluster. Also, we thank VMware for providing academic licenses to run our private cloud environment, being used as the virtualization backend for our GNS3 and VIRL installation, as well as Arista for providing firmware images and support for the switches used in our network.

REFERENCES

[1] S. Reißmann, S. Rieger, and C. Pape, "Using VIRL to improve the scale-out of large virtual network testbeds in higher education," in SIMUL 2017, The Ninth International Conference on Advances in System Simulation, 2017, pp. 29–34.

[2] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown, "Reproducible network experiments using container-based emulation," in Proceedings of the 8th international conference on Emerging networking experiments and technologies. ACM, 2012, pp. 253–264.

[3] M. Pizzonia and M. Rimondini, "Netkit: network emulation for education," Software: Practice and Experience, vol. 46, no. 2, Feb. 2016, pp. 133–165.

[4] S. V. Tagliacane, P. W. C. Prasad, G. Zajko, A. Elchouemi, and A. K. Singh, "Network simulations and future technologies in teaching networking courses: Development of a laboratory model with Cisco Virtual Internet Routing Lab (Virl)," in 2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET). IEEE, 2016, pp. 644–649.

[5] J. Obstfeld et al., "VIRL: The Virtual Internet Routing Lab," SIGCOMM Comput. Commun. Rev., vol. 44, no. 4, Aug. 2014, pp. 577–578. [Online]. Available: http://doi.acm.org/10.1145/2740070.2631463

[6] L. Yan and N. McKeown, "Learning networking by reproducing research results," ACM SIGCOMM Computer Communication Review, vol. 47, no. 2, 2017, pp. 19–26.

[7] V. Bajpai et al., "Challenges with reproducibility," in Proceedings of the Reproducibility Workshop, ser. Reproducibility '17. New York, NY, USA: ACM, 2017, pp. 1–4. [Online]. Available: http://doi.acm.org/10.1145/3097766.3097767

[8] M. Flittner et al., "Taming the complexity of artifact reproducibility," in Proceedings of the Reproducibility Workshop, ser. Reproducibility '17. New York, NY, USA: ACM, 2017, pp. 14–16. [Online]. Available: http://doi.acm.org/10.1145/3097766.3097770

[9] W. Makasiranondh, S. P. Maj, and D. Veal, "Pedagogical evaluation of simulation tools usage in network technology education," World Transactions on Engineering and Technology Education, vol. 8, no. 3, 2010, pp. 321–326.

[10] P. Gil et al., "Computer networks virtualization with GNS3: Evaluating a solution to optimize resources and achieve a distance learning," in 2014 IEEE Frontiers in Education Conference (FIE) Proceedings, Oct 2014, pp. 1–4.

[11] B. Momeni and M. Kharrazi, "Partov: a network simulation and emulation tool," Journal of Simulation, vol. 10, no. 4, 2016, pp. 237–250.

[12] M. A. Qadeer, P. Varshney, and N. H. Khan, "Design and Simulation of Interconnected Autonomous Systems," in 2009 International Conference on Computer Engineering and Technology (ICCET). IEEE, 2009, pp. 270–275.

[13] S. Hemminger, "Network emulation with NetEm," in Linux conf au, 2005, pp. 18–23.

[14] F. Baumgartner, T. Braun, E. Kurt, and A. Weyland, "Virtual Routers: A Tool for Networking Research and Education," SIGCOMM Comput. Commun. Rev., vol. 33, no. 3, Jul. 2003, pp. 127–135.

[15] A. Kayssi and A. El-Haj-Mahmoud, "EmuNET: A Real-time Network Emulator," in Proceedings of the 2004 ACM Symposium on Applied Computing, ser. SAC '04. New York, NY, USA: ACM, 2004.

[16] L. Xu, D. Huang, and W.-T. Tsai, "Cloud-based virtual laboratory for network security education," IEEE Transactions on Education, vol. 57, no. 3, Aug. 2014, pp. 145–150.

[17] T. Wolf, "Assessing student learning in a virtual laboratory environment," IEEE Transactions on Education, vol. 53, no. 2, May 2010, pp. 216–222.

[18] L. Yan and N. McKeown, "Learning networking by reproducing research results," ACM SIGCOMM Computer Communication Review, vol. 47, no. 2, 2017.

[19] HS-Fulda NetLab VIRL topologies. URL: https://gogs.informatik.hs-fulda.de/srieger/git-virl-hs-fulda, 2018-05-20. (2018)

[20] C. Seifert, S. Rieger, and C. Pape, "Realization possibilities for virtual networking labs in higher education courses," in 13 th Annual International Conference on Computer Science and Education in Computer Science 2017 (CSECS 2017), 2017.

[21] C. Pape and C. Seifert, "Adaption and improvement of an industry-developed IP Telephony curriculum," in 7th Annual International Conference on Computer Science and Education in Computer Science, Sofia/Dobrinishte, Jul. 2011, pp. 199–210.

[22] ns-3. URL: https://www.nsnam.org, 2018-05-20. (2018)

[23] OMNeT++ Discrete Event Simulator. URL: https://omnetpp.org, 2018-05-20. (2018)

[24] Packet Tracer - A free network simulation and visualization tool for the IoT era. URL: https://www.netacad.com/courses/packet-tracer, 2018-05-20. (2018)

[25] Mininet - An Instant Virtual Network on your Laptop (or other PC). URL: http://mininet.org, 2018-05-20. (2018)

[26] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks. ACM, 2010, p. 19.

[27] eNSP - Enterprise Network Simulator. URL: http://support.huawei.com/enterprise/en/network-management/ensp-pid-9017384, 2018-05-20. (2018)

[28] GNS3 - The software that empowers network professionals. URL: https://www.gns3.com, 2018-05-20. (2018)

[29] Emulated Virtual Environment Next Generation (EVE-NG) / Unified Networking Lab (UNL). URL: http://www.unetlab.com, 2018-05-20. (2018)

[30] VIRL - Virtual Internet Routing Lab. URL: http://virl.cisco.com, 2018-05-20. (2018)

[31] S. Rieger. Arista vEOS image on VIRL. URL: https://learningnetwork.cisco.com/thread/99040, 2018-05-20. (2018)

[32] K. Spindler et al., "AEQUO: Enhancing the energy efficiency in private clouds using compute and network power management functions," International Journal on Advances in Internet Technology Volume 8, Number 1 & 2, 2015, vol. 8, no. 1 & 2, 2015, pp. 13–28.

[33] HS-Fulda NetLab VIRL utilities. URL: https://gogs.informatik.hs-fulda.de/srieger/virl-utils-hs-fulda, 2018-05-20. (2018)

[34] S. Rieger. GNS3 benchmark script. URL: https://gogs.informatik.hs-fulda.de/srieger/gns3bench, 2018-05-20. (2018)

[35] GNS3 2.0.1 API Documentation - Sample session using curl. URL: http://pythonhosted.org/gns3-server/curl.html, 2018-05-20. (2018)

[36] GNS3. Enhancement: Show where a device is installed. URL: https://github.com/GNS3/gns3-gui/issues/2279#issuecomment-345429116, 2018-05-20. (2018)

[37] P. Bug. virl-scheduler. URL: https://gogs.informatik.hs-fulda.de/pbug/virl-scheduler, 2018-05-20. (2018)