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

Sven Reißmann*, Sebastian Rieger†, 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, christian.pape}@cs.hs-fulda.de

***Abstract*—To build laboratory setups for courses and research projects in higher education dealing with computer networks, virtualized network environments have been continuously gaining momentum over the last years. Depending on the desired practical or theoretical orientation, they can be implemented using different hardware or software solutions. A high practical relevance and functional realism can be achieved using network emulation. However, the 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. This paper describes the solutions used for network emulation and simulation in the network laboratory (NetLab) of Fulda University of Applied Sciences and discusses possibilities to evaluate and improve their scalability and performance.**

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

I. INTRODUCTION

In research and higher education, especially in the area of computer networks, experimental testbeds are a typical requirement. 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 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 providing the required networking components and interfaces, setting up 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 [1] 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 [1], the advantages perceived by using emulation for virtual network testbeds with a high practical relevance is also described in [2], [3] and [4]. Hence, this approach is currently primarily used for higher education courses in the NetLab providing similar advantages that will be discussed in this paper.

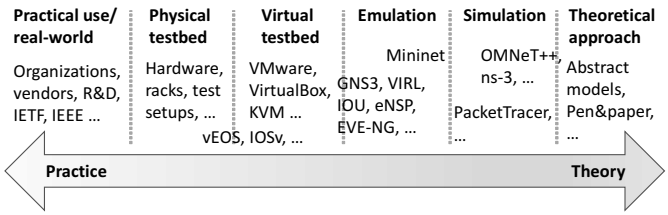


Figure 1. Classification of experimental scientific networking testbeds.

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 paper, we present an analysis of these resource requirements of virtual

 ISBN: 978-1-61208-594-4

network testbeds based on experiences from providing such an environment in the networking laboratory (NetLab) in the Applied Computer Science department of Fulda University of Applied Sciences. 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 paper. Further, we discuss options to improve the scalability of such an environment.

The remaining part of this paper is laid out as follows. Section II introduces software products available for experimental networking testbeds based on the classification and examples shown in Figure 1. The following Section III discusses related work in this area. Based on our experiences during the implementation of these solutions, in Section IV we present our selection of tools for creating a scalable platform for virtual networking testbeds. Section V gives examples for emulated network topologies used in higher education courses in the NetLab including a description of the implementation. Next, an evaluation of the scalability and performance of the environment is presented in Section VI. Finally, a conclusion and future work can be found in Section VII.

II. Virtual Environments for Network Testbeds

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 [1], 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) [5]. 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 practical 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 [6] or OMNeT++ [7] is mentioned in some lectures in master's programs, but are not currently used for practical relevant experiments in the lab.

Practical training for the previously mentioned CCNA certification includes the use of Cisco Packet Tracer [8]. 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 that 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 [9], mentioned in [10] and [1], is also provided in our NetLab, where it is mainly used for experiments in the SDN area. While the resource requirements of Mininet are low, due to a container-based approach, individual topologies require 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.

III. Related Work

Services in today's cloud-driven infrastructures are based on sophisticated network topologies that interconnect various network and server components. The simulation and emulation of such network topologies for teaching higher education classes is subject of current research. In [3], a virtual environment based on VIRL is compared to physical setups using CCNA pods and network simulation software like Cisco Packet Tracer. Furthermore, the cost, setup requirements and limitations are estimated and reviewed. The use of VIRL in the area of research and education is also discussed in [4]. 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 [2]. The goal of this emulation environment software is to setup networking experiments at low cost and with little effort. A framework for reproducible container-based emulation of networking experiments is presented in [1].

IV. Rationale for the Selection of VIRL

Over the last years, eNSP [14], GNS3 [15], EVE-NG [16] (formerly UNetLab) and VIRL [17] have been deployed and evaluated as a solution for realistic emulation of networking environments in the NetLab. Currently, VIRL is the most promising option used in the several courses. The advantages

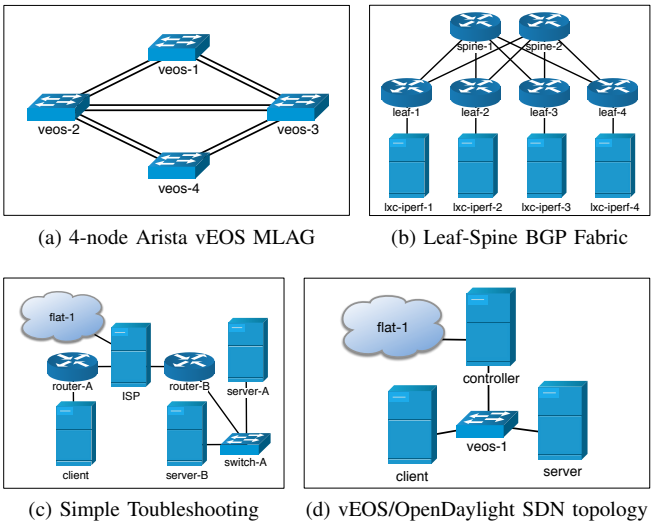
 ISBN: 978-1-61208-594-4

of VIRL are related to the findings in [1] and [4]. The functional realism of VIRL is extended compared to the other alternatives as it allows to use official 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-independent (i.e., from within the NetLab or from at home using private PCs and laptops, which is specifically important for students). 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 specially tailored testbeds for the use in research and education. 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 in the area of research and education will be discussed in the following sections of the paper.

V. Examples for Emulated Network Topologies in our Networking Laboratory (NetLab)

Refer to Figure 2 for VIRL-based 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 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 realized 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.

An example of a much simpler network topology is shown in Figure 2c. Students in the bachelor program troubleshoot



(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.

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. 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 mentioned in this section are managed using Git and can be downloaded from [18]. 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 [19]. For example, to allow the operation of MLAG, it was necessary to modify the *base_mac* in order to prevent clashes of MAC address generated by vEOS with locally generated KVM MAC addresses [19].

VI. 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. Hence, 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.

 ISBN: 978-1-61208-594-4

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 [20]. 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 our VMware vSphere 6.5 cluster in which each of the four VMs is bound to a separate physical ESXi host by DRS constraints. 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, respectively. As discussed in [1], 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 benchmarking scenarios were performed at night in the semester break to ensure minimum load 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 the 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 that establishes WebSocket connections to the serial consoles of the nodes running on the VIRL hosts.

A schematic representation of the VIRL environment is depicted in Figure 3. 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.

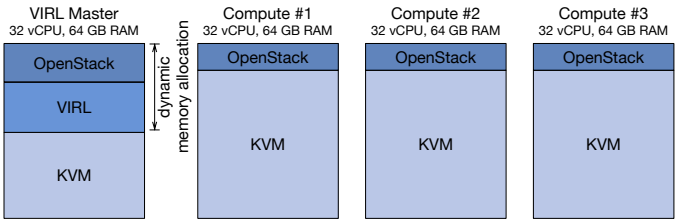


Figure 3. Schematic view of the environment's memory usage.

In Section VI-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 [21].

B. Scalability Evaluation

The results from our previously explained test case are illustrated in Table I and Figure 4. 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 (Figure 3). Hence, a stable execution was not possible with a single host, even with memory overprovisioning enabled (Figure 4a).

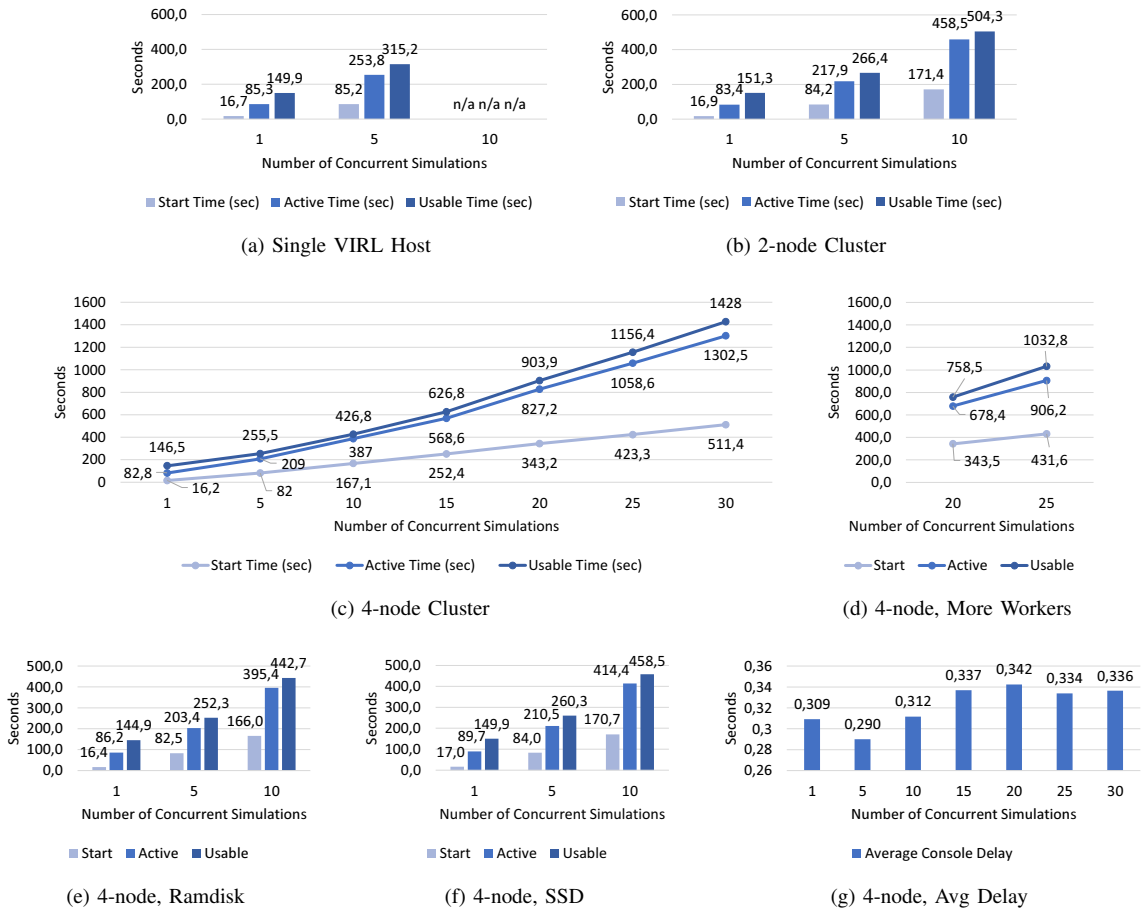
TABLE I. RELATIVE CHANGE IN *Start Time*, *Active Time* AND *Usable Time* DEPENDENT ON THE NUMBER OF TOPOLOGIES.

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 4b 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 4c). 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 and increase the difference between *Active Time* and *Usable Time*. The curve for *Usable Time* has a larger 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

 ISBN: 978-1-61208-594-4



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

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 3). 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 4d 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 4g. 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 und VIRL management processes, as well as the boot process of the vEOS instances, which utilizes the assigned vCPU to 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 a 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 only supported on the controller in VIRL, hence we needed to manually configure it on the compute nodes. Figure 4e 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 4f 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 would assume that an improvement is observable when the I/O load of the VMs increases as a result of more complex topologies.

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 paper, 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. 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 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 resources management, as well as the underlying network infrastructure.

By the time of writing, new major versions of the previously mentioned software projects GNS3 and EVE-NG were released, both with valuable new features. As an example, EVE-NG added support for multiple simultaneous users who can work on the same topology, which is a requirement to qualify for educational use. Still, both projects do not provide a way to use Cisco operating system images in conformance with the license requirements, however, other network operating systems like Arista vEOS are available. In the future, we will setup a new lab environment with these software projects to evaluate the suitability and scalability for our use-cases. 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.

References

[1] 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.

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

[3] 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.

[4] 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

[5] 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.

[6] ns-3. URL: https://www.nsnam.org, 2017.06.07. (2017)

[7] OMNeT++ Discrete Event Simulator. URL: https://omnetpp.org, 2017.06.07. (2017)

[8] Packet Tracer - A free network simulation and visualization tool for the IoT era. URL: https://www.netacad.com/about-networking-academy/packet-tracer, 2017.06.07. (2017)

[9] Mininet - An Instant Virtual Network on your Laptop (or other PC). URL: http://mininet.org, 2017.06.07. (2017)

[10] 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.

[11] B. Momeni and M. Kharrazi, "Partov - a network simulation and emulation tool." J. 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] eNSP - Enterprise Network Simulator. URL: http://support.huawei.com/enterprise/en/network-management/ensp-pid-9017384, 2017.06.07. (2017)

[15] GNS3 - The software that empowers network professionals. URL: https://www.gns3.com, 2017.06.07. (2017)

[16] Emulated Virtual Environment Next Generation (EVE-NG) / Unified Networking Lab (UNL). URL: http://www.unetlab.com, 2017.06.07. (2017)

[17] VIRL - Virtual Internet Routing Lab. URL: http://virl.cisco.com, 2017.06.07. (2017)

[18] HS-Fulda NetLab VIRL Topologien. URL: https://gogs.informatik.hs-fulda.de/srieger/git-virl-hs-fulda, 2017.06.07. (2017)

[19] S. Rieger. Arista vEOS image on VIRL. URL: https://learningnetwork.cisco.com/thread/99040, 2017.06.07. (2017)

[20] 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.

[21] HS-Fulda NetLab VIRL Utilities. URL: https://gogs.informatik.hs-fulda.de/srieger/virl-utils-hs-fulda, 2017.06.07. (2017)

 ISBN: 978-1-61208-594-4