LightVN: A Light-Weight Testbed for Network and Security Experiments

Quamar Niyaz, Weiqing Sun, Rao Xu, and Mansoor Alam

The University of Toledo, Toledo, OH-43606, USA
{quamar.niyaz, rao.xu}@rockets.utoledo.edu, {weiqing.sun, mansoor.alam2}@utoledo.edu

***Abstract*—Testbeds are essential in teaching and research on computer networks and security. However, designing and deploying a testbed is an expensive task. Various virtualization technologies make this task easier by enabling the creation and deletion of virtual machines (VMs) and their networks in less time on a single physical infrastructure. However, these VM based testbeds are heavy-weight. In this paper, we propose a light-weight testbed architecture for performing network and security experiments. The testbed, termed as *LightVN*, supports multiple virtual networks set-up on a single physical network infrastructure, while maintaining the isolation between them. Based on network and OS-level virtualization and using open source tools, the testbed is cost-effective, scalable, and manageable. Finally, we evaluate the performance of our testbed set-up in the low-cost commodity server. We observe that it can simultaneously support a large number of virtual networks for different user groups.**

***Keywords*—Network and security, testbed, light-weight virtualization, open vSwitch, LXC.**

I. Introduction

Teaching and research on computer networks and security need experimentation to understand the concepts or proving new ideas. In order to achieve this, instructors or researchers either use simulation tools or real testbeds. Simulation-based approach provides a cost-effective, scalable, and easy-to-reconfigure environment. However, the interfaces and experiences that simulation tools offer are far from real-world experiences. In contrast, physical testbeds provide real-world experiences as they carry actual network traffic and mimic real networks. Experimentation on a real testbed helps in sound understanding of the concepts and provides strong claims in support of new ideas.

However, various challenges arise with the use of real physical testbeds. First, they need a reasonable amount of investment. In a shared environment, different user groups share the same testbed. Network configurations, e.g. firewall settings, of a user group can affect the traffic of others in this confined environment. In addition, many of the system configurations require administrator authority. This privilege gives an opportunity to malicious user groups to perform illegitimate activities on others' file systems and system settings. A separate testbed for each user group helps in carrying out experiments without any effect of suspected malicious activities of others. However, providing a separate testbed to each user group is expensive and needs much effort in an academic set-up.

The second challenge with these testbeds is the security threat to the public network. System malware or viruses may escape from the testbed, enter into the public network from the testbed environment and cause adverse effects to a large user base. Isolation of the testbed from the public network reduces the security threats. However, it makes remote accessing of the testbed, from the public network, difficult.

With the advent of various virtualization technologies, such as Xen, VMware, and Virtual Box, the task of setting up testbeds has become easier as compared to the deployment of an actual physical testbed. This convenience motivated many researchers from different institutions to develop testbeds using virtualization. These testbeds consist of inter-networked virtual machines (VMs). We provide a detailed discussion of these testbeds in Section II. Although virtualization helped greatly in deploying the testbeds, most of the testbeds are built on full-virtualization. In such an environment, a VM is a complete system with its kernel separated from the host. It runs on top of a software called a hypervisor or virtual machine monitor (VMM) and indirectly access the host resources via the hypervisor or VMM [15]. The downside of full-virtualization is that VMs are heavy-weight and utilize system resources heavily. This heavy-weightiness puts constraints on the number of VMs that a physical server can accommodate. Therefore, it restricts the number of testbeds that can be deployed on it.

We propose a light-weight testbed, termed as *LightVN*, to overcome the scalability issue of previously deployed testbeds. It hosts multiple virtual networks over a single underlying network infrastructure. We build LightVN using *Linux Containers* and *Open vSwitch*. Both these tools, available under the open source license, make the architecture light-weight as compared to other testbeds. Although the virtual networks share a single network infrastructure, we make sure that the traffic and settings of these networks do not interfere with each other. LightVN redirects different user groups to different virtual networks on the same underlying network based on the user group identity. Each user group gets a different instance of the underlying infrastructure and has the illusion that it owns a dedicated testbed. The proposed architecture is cost-effective, scalable and provides a user-friendly management environment to the administrator.

The rest of the paper is organized as follows. Section II presents the related work. We present the design and implementation of *LightVN* in Section III and Section IV. Section V evaluates the performance of the testbed when it hosts multiple virtual networks simultaneously. We discuss the limitation of *LightVN* in Section VI and concludes our paper with future work directions in Section VII.

II. Related Work

Various virtualization technologies ease the task of deploying testbeds for network and security related experiments. Many institutions made use of these technologies and developed testbeds for their academic courses. They used full-virtualization and developed additional middle-ware for their testbeds. V-NetLab provides a virtual network lab set-up with a few VMs for a user group [6]. An administrator or user provides a network configuration using text-based script that is lexically and syntactically analyzed for its correction. If the given network specification is correct a virtual network is created corresponding to that specification. It uses data-link layer virtualization in order to isolate the logical networks from the public network [12]. SOFTICE consists of a load-balanced cluster of PCs and uses Warewulf [2]. In this architecture, the user connects to the master node, which also acts as a gateway, and is redirected to one of the PCs of the cluster. In that PC, the user provides a network configuration file. The Manage Large Networks (MLN) [13] utility creates the virtual network from the configuration file using UML/Xen on that node. Xen Worlds is deployed at Iowa State University for the Information Assurance program [1]. It provides a virtual lab environment to on-campus as well as off-campus students. It uses Xen server at its core and pdmenu, a menu driven SSH client interface to make it accessible to the client from remote. V-Lab is a cloud based flexible and re-configurable virtual network laboratory set-up [17]. It also uses Xen server, and VLAN supported physical switch to isolate the virtual networks. It provides a GUI-based web interface to the instructors to configure a network testbed. The students can access the virtual network system via SSH, VNC, or RDP. VNLab is another approach that uses Microsoft Virtual Server for implementing their virtual laboratory [3].

Unlike these approaches, we use light-weight OS-virtualization and Open vSwitch to accommodate a large number of virtual networks on the underlying infrastructure. Mininet, a container based emulation tool, also uses light-weight virtualization by exploiting the features of the network namespace and Open vSwitch [7]. However, it is intended for prototyping the software-defined networks and providing an emulation environment for them. In addition, Mininet virtual hosts share the host file systems; therefore, system configuration changes for the host can be reflected in all virtual hosts.

III. Tools Overview

As discussed, we use Linux Containers and Open vSwitch for the implementation of our testbed. Both of them are available as open source and implemented over the Linux platform. A brief overview of these tools are as follows:

A. Linux Containers

Linu**X** **C**ontainers (**LXC**) are based on OS-level virtualization [8]. Although there are other OS-level virtualization tools in Linux, such as OpenVZ, and Linux-V server, we use LXC since it comes with the Linux kernel. This makes easier to work-around with LXC as compared to other tools. LXC allows to run a complete instance of a Linux system without using any emulation of real physical hardware, such as hypervisor or VMM. LXCs are designed to run various network services, such as web, DNS, or mail server, or multiple instances of them on a single host. Each LXC runs in a secure and isolated environment. These features of LXCs avoid the need of dedicated systems for hosting the network services separately. Therefore, it helps in cost-effective utilization of network resources and infrastructures instead of having various under-utilized resources. LXC shares the host kernel and directly executes over it. It makes the LXC light-weight and better in terms of performance as compared to other virtualization tools based on different techniques including para-virtualization and full-virtualization [11]. An LXC relies on the `cgroup` and `namespace` isolation which became part of the mainline Linux kernel. Using `cgroup` enables the administrator to limit CPU, memory usage and disk I/O by various LXCs running on a single host. LXC provides many features, but its separate file system, network isolation, and root privilege separation characteristics make it a suitable candidate to use it in our testbed.

B. Open vSwitch

Open **vS**witch (**OVS**) is an open source software switch. It implements an Ethernet switch as its most basic functionality and can serve as a production quality and multi-layer virtual switch providing network automation with the help of programmatic extension at its extreme [10]. Its design aim was to be used in a virtualized server environment to forward traffic between different VMs on the same physical host or the same network. It can run on any Linux-based virtualization platform, including Virtual box, Xen, Xen Cloud Platform, Xen Server and became part of the mainline kernel. However, it can be ported to other environments as well. It can operate both as a software switch running within the virtual machine hypervisors and as the control stack for programmable hardware switches. It has been integrated into many virtual management systems including OpenStack, OpenNebula, OpenQRM. It has as many features as supported by the traditional hardware switches, although its support for fully functional Layer-2 switch, VLAN with trunking, and Generic Routing Encapsulation (GRE) tunnelling make it a suitable tool for our testbed.

IV. LightVN Testbed: Architecture, Management, & Access

LightVN offers multiple instances of the underlying network infrastructure as separate virtual networks to different user groups. The design of such an architecture must meet the following requirements:

- Each user group is oblivious with the presence of other virtual networks that co-exist with its virtual network on the same underlying infrastructure.
- Traffic of one virtual network must be isolated from other networks.
- Firewall, IDS, or other network configurations of a virtual network should not affect traffic of other networks.
- The interface for accessing the virtual networks should be user-friendly and easily accessible by remote access tools, such as SSH and VNC.

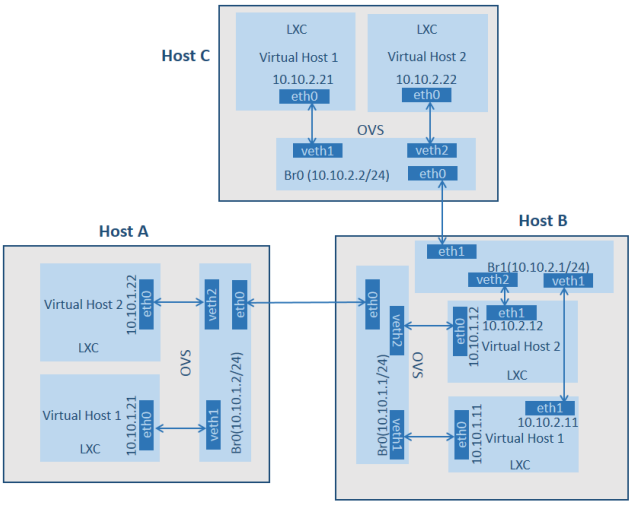

Fig. 1. LightVN testbed with two virtual networks, each having three virtual hosts, hosted over a single network infrastructure of three physical hosts. Host B acts as a router with two network interfaces.

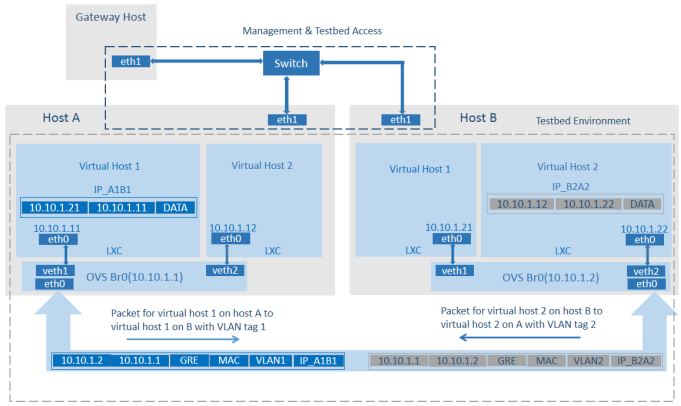

Fig. 2. GRE tunnelling between two hosts, both hosts are tunnelled with the remote IP of each other. Management interfaces are not included for GRE tunnelling.

- Administrator of the testbed should not have too much burden to manage the virtual networks for all user groups. All network management and configurations should be performed with the help of an automated system.

A. Architecture

We create an LXC on an underlying host of the infrastructure for each virtual host. Each virtual host is assigned to a particular user group on an underlying host. The underlying host is either a physical machine or a virtual machine, although we refer it as *physical host* in the rest of the paper. The network configuration of the virtual host is similar to the physical host. For each network interface in the physical host, except the management network interface, we create an OVS bridge and associate it with the corresponding network interface. We use the management network interface to connect physical hosts with the gateway host. The user groups and the administrator access the testbed through the gateway host. The administrator accesses the physical hosts in order to manage/monitor virtual hosts for different user groups. The user groups access their virtual hosts console on the physical hosts by accessing them using the management network interface. We assign each virtual host a virtual network interface corresponding to a network interface of the physical host. A virtual interface exists in a pair, one end of which is associated with the virtual host (LXC), and the other end is hooked into the OVS bridge.

In order to isolate the traffic of a virtual network from the others, we use the VLAN feature of OVS. We assign all the virtual hosts of a user group a unique VLAN id on all the physical hosts. Assigning a unique VLAN id to the virtual hosts of a user group creates a virtual network for that group and isolates its traffic from others [16]. We use GRE tunnelling to confine the virtual networks traffic within the testbed [4]. We create a GRE port on each OVS bridge and associate it with the remote IP of the neighbouring host's network interface for the same subnet. In this way, we successfully separate the testbed traffic from the public network.

Figure 1 shows the architectural diagram of the LightVN testbed for an underlying infrastructure of three physical hosts. We have not shown the connectivity of the testbed with the gateway host; therefore, management interfaces are not shown in the figure. The testbed hosts two virtual networks as an example. The *virtual host 1* on each physical host form the virtual network for one user group. In the same manner, all the *virtual host 2* form a different virtual network for a different user group. We assign different VLAN to each user group by assigning a unique VLAN tag to each port corresponding to the virtual network interfaces of all the virtual hosts of a user group. Figure 2 shows the GRE tunnelling between two neighbouring hosts. It does not include the management network interfaces due to which the virtual networks traffic are confined within the testbed environment and remain isolated from the public network.

B. Testbed Management

LightVN uses an automated system to manage and monitor virtual networks. It has a gateway host which acts as an administration host. It also acts as a gateway for user groups to access their virtual networks. The gateway host has two network interfaces; one connects it with the public network, and the other connects it with the management network of the LightVN. Figure 3 shows the management architecture of the LightVN. Currently, we have designed the management interface based on the assumption that every virtual network is an instance of the underlying network. We have a pre-configured LXC on every physical host which is used as a template in the creation of a new virtual host on that host. The number of network interfaces on the template LXC is the same as the number of interfaces on the physical host excluding the management interface. We install various network utilities and services on the LXC template. With the help of a start-up script, OVS switches are configured for a large range of VLAN tag port on each host. We maintain an IP range for different subnets on every physical host similar to the DHCP service. It helps in assigning IP addresses to the virtual network interfaces of the virtual host. The subnets of the virtual hosts and the underlying physical host are the same. Every physical host runs a daemon server `vnetd` as a management application. The administrator in the gateway host connects

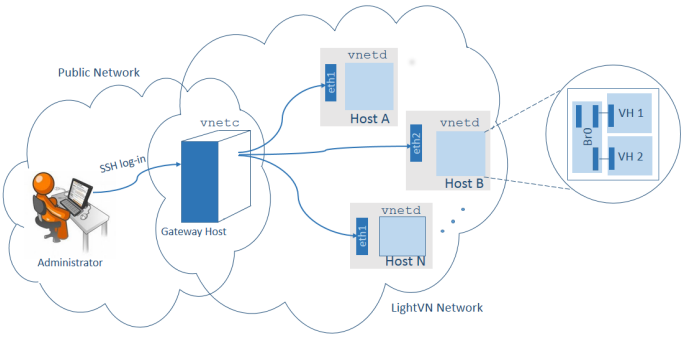

Fig. 3. The administrator and user groups access the testbed using the Gateway host from a public network. The administrator accesses the physical hosts using `vnetc` client application in the figure.

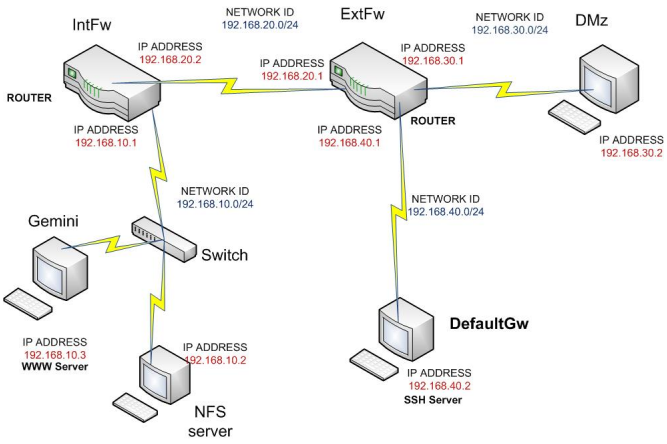

Fig. 4. A network topology used for course assignments in V-NetLab [6].

with the `vnetd` server using a `vnetc` client application as shown in Figure 3. Both `vnetd` and `vnetc` applications are written in Python with few additional bash scripts. The administrator executes the client application for performing the following three actions in the current implementation:

- *Add a virtual network*: The administrator executes the `vnetc` client application using the command `vnetc add group1`. The `vnetc` client establishes a connection with `vnetd` server on each host. The `vnetd` server on each host parses the arguments for adding a virtual host corresponding to a user group. It creates a user group called `group1` and a virtual host corresponding to the user group using the LXC template present in the host. It uses the `lxc-clone` feature to create a new virtual host. After that, it assigns IP to each interface of the virtual host using the available IPs from the managed IP range list. The host name of each virtual host is similar to the physical host to give an illusion to every user group that it has its own dedicated testbed. The `vnetd` server puts an entry for the new virtual host id and its corresponding user group in the `user.conf` configuration file. This entry helps in redirecting the user group to its virtual host on a physical host after accessing that host.
- *Delete a virtual network*: To delete an existing virtual network of a user group, the administrator host executes the `vnetc` client application using the command `vnetc del group1`. The `vnetd` server on each host parses the arguments to delete a user group. It checks the `user.conf` file for the virtual host corresponding to the given user group `group1`. It deletes the virtual host and the user group account from the host. It uses the `lxc-destroy` feature to delete a virtual host. Then, it removes the entry for the user group and its corresponding virtual host from the `user.conf` file.
- *Monitor a virtual network*: To monitor the virtual network of a user group, the administrator executes the client application using the command `vnetc monitor group1`. The `vnetd` server parses the arguments and returns the running status of the virtual host with its IP to the client application using the `lxc-ls` feature. The client displays the status, as received from the `vnetd` servers, for all the virtual hosts corresponding to the given user group.

C. Remote Access

The user group accesses the testbed using the gateway host via an SSH client similar to the administrator log-in as shown in Figure 3. We give access to all user groups through the gateway host without any administrator privilege on the host. The user group accesses its virtual hosts via SSH from the gateway host. Every physical host has a log-in bash script that checks the user group identity during the SSH login. If the user is not an administrator then the script checks the `user.conf` file and finds the corresponding virtual host for the user group and automatically redirects the user group to that virtual host console. After log-in to the virtual host console, the user can set-up its network services and configure the network without affecting the other user groups. An administrator gets direct access to the physical host for managing and monitoring the virtual hosts and their users on it using the `vnetc` client application.

V. System Performance

We evaluate the performance of LightVN testbed by porting V-NetLab [6] on it. V-NetLab is used by the instructors of many institutions for carrying out various experiments as part of teaching in computer network and security courses. In this architecture, a group of 4-5 students is assigned a testbed of 6-8 VM hosts as shown in Figure 4. The testbed models an enterprise network topology. To accommodate the new user group, 6-8 VMs are created. In LightVN, instead of creating a new testbed of VMs for a particular user group, we create a single testbed of 6-8 VMs and create light-weight instance of this testbed for every new user group. We use a Dell PowerEdge 2950 server with Intel Xeon CPU E5420 @ 2.50 GHz and 16GB RAM. There are 8 CPU cores with 2.493 GHz. It runs VMWare ESXi host to provide the virtualization platform. We create six host VMs (physical hosts), and one Gateway host VM with the configuration of 4GB RAM and 4 CPU cores on this server. We evaluate our testbed performance by varying the number of virtual networks from 2 to 20 hosted on the same underlying infrastructure. For each virtual network, there is a virtual host on every physical host. We

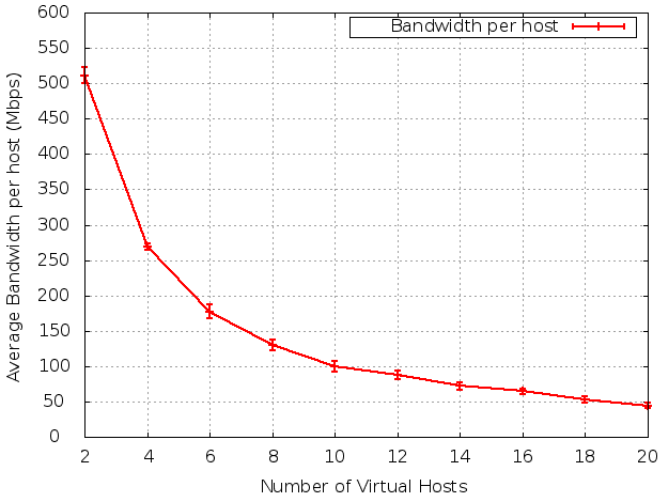

Fig. 5. Average available bandwidth with a standard deviation v/s number of virtual hosts on a physical host corresponding to the number of user groups/virtual networks.

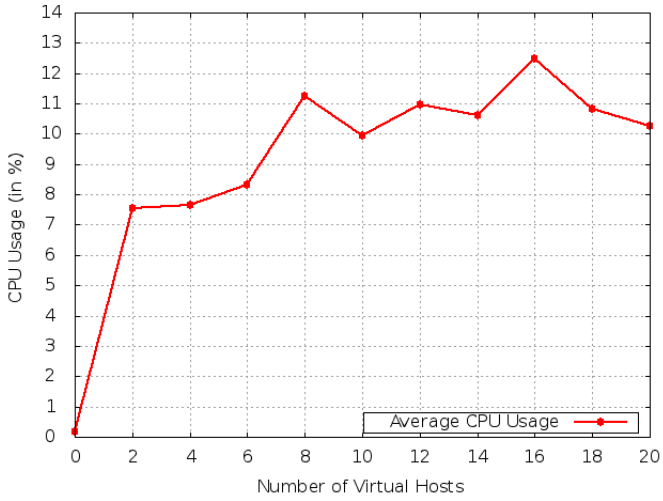

Fig. 6. Average CPU Usage (in %) of the physical host v/s number of virtual hosts corresponding to the number of user groups/virtual networks.

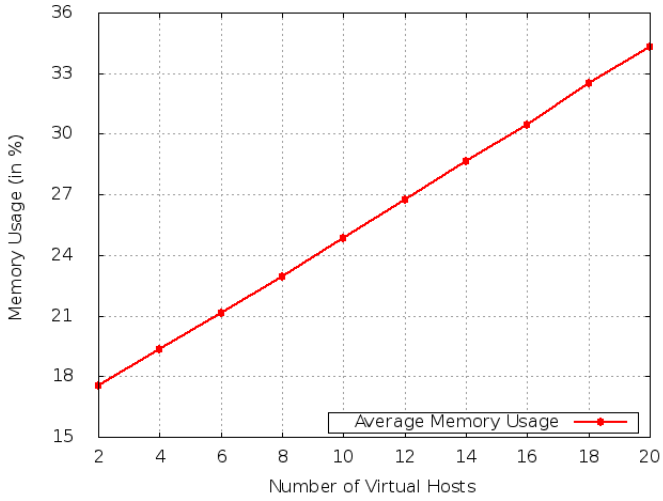

Fig. 7. Average Memory Usage (in %) of the physical host v/s number of virtual hosts corresponding to the number of user groups/virtual networks.

measured available network bandwidth to each virtual host, CPU and memory usage of the physical host running the virtual hosts, and network latency for a maximum path length in the topology. Following are our observations:

- Network Bandwidth: To evaluate the average network bandwidth for each virtual host, we run `iperf` [5] utility with UDP client and server on two virtual hosts of each virtual network, one runs on the *Gemini* host and the other runs on the *DMZ* host as shown in Figure 4. We show the result for available bandwidth with respect to the number of virtual hosts on the physical host *Gemini* in Figure 5. We observed that with the increase of the number of virtual hosts, available bandwidth to them decreases from 511.5 Mbps for two virtual networks to 44.5 Mbps for 20 virtual networks. And it reflects the fair share of total available bandwidth.
- CPU Usage: We used `sysstat/sar` [14] tool utility for measuring the average CPU and memory usage of the physical host while the virtual hosts on it were running the `iperf` utility. Figure 6 shows the result for CPU usage. We observed that, there is an increase in CPU usage with the increase in the number of virtual hosts. However, the average percentage usage went upto 12.5% (maximum) that indicates that the physical host is not over-utilized.
- Memory Usage: We used the same `sysstat/sar` tool and followed the same procedure as we did for evaluating the CPU usage. Figure 7 shows the result for memory usage of the physical host. We observed that there is a linear increment in the memory usage with the increase in the number of virtual hosts. The memory usage of the physical host reached 34.5% for 20 virtual hosts from 17% for two virtual hosts.
- Network Latency: In V-NetLab set-up, the maximum path length between any two hosts is 3 hops. We measured the average round-trip latency for this path on every virtual host when all of them were performing the network I/O on this maximum length path. Figure 8 shows the result for average round-trip latency. We observed that round-trip latency increases with the increase of the number of virtual hosts on the end hosts, however the values are not too high. The maximum average round-trip latency for each virtual host is 6.4 ms for 20 virtual hosts.

On the same Dell server, we were able to run 20 virtual networks simultaneously without noticeable performance degradation using LightVN as compared to 8 virtual networks maximally based on the V-NetLab platform.

VI. Discussion

With the help of LightVN, we aim to provide a light-weight virtualized testbed environment. There are a few important issues, however, that need a discussion. First, LXC offers virtualized environment around Linux based OS. Every user group

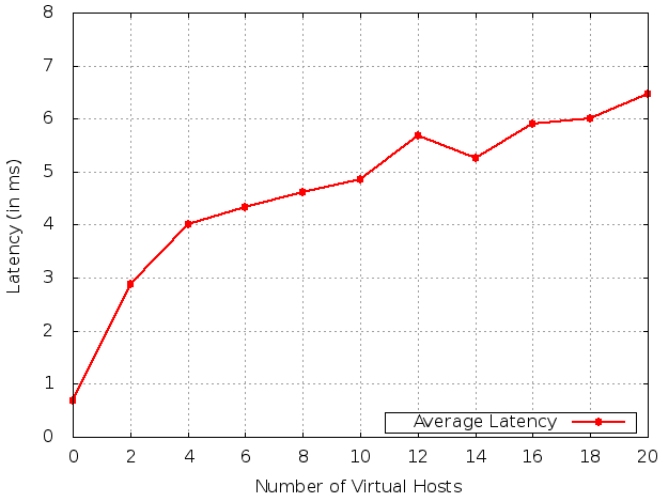

Fig. 8. Average Round Trip Latency for 3-hop path v/s number of virtual hosts corresponding to the number of user groups/virtual networks.

gets its virtual network with Linux OS instances. However, we do not consider it as a limitation for the LightVN as Linux is an open-source, available with many distributions, customizable, and provides a close interaction to the user with the OS. Because of these properties, it is widely used in academics for research and teaching.

The second issue with OS-based virtualization, as used in LXC, is that it enables the LXC based virtual host to share the host system kernel. If the host kernel contains any vulnerable system calls, the malicious virtual host can exploit them as well. It would give the malicious virtual host access to the host system and other virtual hosts. However with the use of AppArmor, SELinux, SMack, and Seccomp features these threats can be minimized [9]. Sharing the host system kernel restricts an LXC to host network services that need to run on kernel-space, such as, NFS server. In order to mitigate this issue, one might use user-space NFS server instead.

Finally, in our implementation of LightVN, we offer an instance of the underlying network infrastructure to a user group. One might consider this model scalable with less flexibility. However, by using this setup, we can easily provide the testbed to user groups for a particular network and security course. We assume that the same set of experiments are assigned to each user group during the coursework. With the presence of tools, such as Manage Large Networks (MLN), or the network configurator of V-NetLab, creating an underlying infrastructure for the LightVN is not a cumbersome task. We can easily create an underlying infrastructure for a LightVN testbed for different network topologies with the help of these tools. Therefore, we can emphasize that LightVN maintains the flexibility as well as scalability.

VII. Conclusions and Future Work

In the paper, we discussed, *LightVN*, a light-weight virtual network testbed architecture for supporting networks and security experiments. This testbed aims to help in deploying a large number of virtual networks over an underlying infrastructure with limited resources. It also makes the previously deployed testbed more scalable. We have ported V-NetLab architecture on it and evaluated the testbed performance with a reasonable number of virtual networks hosted simultaneously. We observed that the testbed can accommodate all these virtual networks well with a reasonable usage of resources. We asked a few graduate students who used V-NetLab previously for their course work to use this testbed. They were asked to perform the same lab assignments as given in their course work, such as topology discovery, network services configuration, traffic monitoring, and firewall/IDS configuration. We found that their user-experiences were almost the same as their experiences for the V-NetLab. We plan to use *LightVN* testbed for various courses in network and system courses in the upcoming semesters. In addition, we will invite researchers to perform network and security experiments using the testbed. Based on the feedback from a large group of users, we will further improve LightVN.

References

[1] Benjamin R. Anderson, Amy K. Joines, and Thomas E. Daniels. Xen Worlds: Leveraging Virtualization in Distance Education. In *Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education*, ITiCSE '09, pages 293–297, New York, NY, USA, 2009. ACM.

[2] William D. Armitage, Alessio Gaspar, and Matthew Rideout. A UML and MLN Based Approach to Implementing a Networking Laboratory on a Scalable Linux Cluster. *J. Comput. Sci. Coll.*, 23(2):112–119, December 2007.

[3] Dalibor Dobrilovic, Vesna Jevtic, Zeljko Stojanov, and Borislav Odadzic. Usability of Virtual Network Laboratory in Engineering Education and Computer Network Course.

[4] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina. Generic Routing Encapsulation, RFC 2784.

[5] Iperf The TCP/UDP Bandwidth Measurement Tool. https://iperf.fr/.

[6] Kumar Krishna, Weiqing Sun, Pratik Rana Rana, Tianning Li, and R. Sekar. V-NetLab: A Cost-Effective Platform to Support Course Projects in Computer Security. In *9th Annual Colloquium for Information Systems Security Education*, CISSE'05, 2005.

[7] Bob Lantz, Brandon Heller, and Nick 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*, Hotnets-IX, pages 19:1–19:6, New York, NY, USA, 2010. ACM.

[8] Linux Containers. https://linuxcontainers.org/.

[9] LXC-Security. https://help.ubuntu.com/14.04/serverguide/lxc.html.

[10] Open vSwitch. http://openvswitch.org/.

[11] Stephen Soltesz, Herbert Pötzl, Marc E. Fiuczynski, Andy Bavier, and Larry Peterson. Container-based Operating System Virtualization: A Scalable, High-performance Alternative to Hypervisors. In *Proceedings of the 2Nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*, EuroSys '07, pages 275–287, New York, NY, USA, 2007. ACM.

[12] Weiqing Sun, Varun Katta, Kumar Krishna, and R. Sekar. V-NetLab: An Approach for Realizing Logically Isolated Networks for Security Experiments. In *Proceedings of the Conference on Cyber Security Experimentation and Test*, CSET'08, 2008.

[13] The MLN Project- Complex Virtual Machine Management Made Easy. http://mln.sourceforge.net/.

[14] Sysstat Tool. https://github.com/sysstat/sysstat.

[15] Understanding Virtualization, Paravirtualization, and Hardware Assist. http://www.vmware.com/files/pdf/vmware_paravirtualization.pdf.

[16] Virtual-LAN. http://en.wikipedia.org/wiki/virtual_lan.

[17] Le Xu, Dijiang Huang, and Wei-Tek Tsai. V-lab: A Cloud-based Virtual Laboratory Platform for Hands-on Networking Courses. In *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '12, pages 256–261, New York, NY, USA, 2012. ACM.