2018 IEEE 32nd International Conference on Advanced Information Networking and Applications

# NIEP: NFV Infrastructure Emulation Platform

Thales Nicolai Tavares*, Leonardo da Cruz Marcuzzo*, Vinícius Fulber Garcia*, Giovanni Venâncio de Souza‡, Muriel Figueredo Franco†, Lucas Bondan†§, Filip De Turck§, Lisandro Zambenedetti Granville†, Elias Procópio Duarte Junior‡, Carlos Raniery Paula dos Santos*, Alberto Egon Schaeffer-Filho†
*Federal University of Santa Maria
{tntavares,lmarcuzzo,vfulber,csantos}@inf.ufsm.br
†Federal University of Rio Grande do Sul
{lbondan,mffranco,alberto,granville}@inf.ufrgs.br
‡Federal University of Paraná
{gvsouza,elias}@inf.ufpr.br
§INTEC – Ghent University
{filip.deturck}@ugent.be

***Abstract*—Network Functions Virtualization (NFV) presents several advantages over traditional network architectures, such as flexibility, security, and reduced CAPEX/OPEX. However, virtualizing network functions usually executed on specialized hardware (*e.g.*, firewall, DPI, load balancer) and employing innovative technologies (*e.g.*, OpenFlow, P4) increases the challenges of designing, testing, and deploying network infrastructures and services. Although platforms for prototyping NFV environments have emerged in recent years, they still present limitations that hinder the evaluation of specific NFV scenarios, such as fog computing and heterogeneous networks. In this paper, we present NIEP: a platform for designing and testing NFV-based infrastructures and Virtualized Network Functions (VNFs) through the integration of a well-known network emulator (Mininet) and a novel platform for Click-based VNFs development (Click-on-OSv). NIEP provides a complete NFV emulation environment, allowing network operators to test their solutions in a controlled scenario prior to deployment in production networks. As main advantages, NIEP allows the emulation of heterogeneous scenarios, which can be easily migrated to production environments. An experimental scenario is defined to analyze NIEP's performance in terms of VNFs boot time and throughput. Further, NIEP's advantages and shortcomings are discussed and compared to existing emulation platforms.**

***Index Terms*—NFV, SDN, infrastructure emulation, VNF design**

## I. Introduction

Network Functions Virtualization (NFV) is driving a paradigm shift in telecommunications by avoiding ossification and introducing innovation in the network core [1]. NFV transforms the way in which operators design and manage networks by employing virtualization technology to consolidate specialized network equipment onto commodity servers. By moving the processing of packets from dedicated middleboxes to Virtualized Network Functions (VNFs) running on commercial off-the-shelf (COTS) servers, NFV enhances flexibility and scalability to create innovative services while reducing CApital and OPerational EXpenditure (CAPEX and OPEX) [2].

Both academia and industry have been carrying out efforts to evolve and promote NFV. These efforts include, for example, the development of novel NFV architectures, systems, and applications [3]. One of the main challenges for developers and researchers in NFV is the evaluation of, for example, the performance and behavior of VNFs prior to their actual deployment in production networks. Difficulties to perform tests include infrastructure limitations and unavailability of actual NFV environments. As such, solutions to emulate NFV scenarios are important to help in the process of designing, testing, and evaluating VNFs.

The use of emulation to evaluate applications before deployment has been widely applied in computer networks in the past [4] [5]. In the same way, by introducing emulation environments with support for both NFV infrastructures and the Management and Orchestration (MANO) framework, developers and researchers are better instrumented to improve their VNFs, while not running the risk of compromising the production environment. However, despite the inherent benefits, solutions for NFV emulation are scarce, limited (*e.g.,* due to low portability or lack of support for heterogeneous environments), not intuitive, and involve a steep learning curve before they can be fully adopted.

In this paper, we present NFV Infrastructure Emulation Platform (NIEP) [1], a novel platform based on Click-on-OSv [6] and Mininet [7] that emulates diverse NFV scenarios and allows the evaluation of VNFs. NIEP allows operators to rapidly create heterogeneous NFV emulated scenarios. All created scenarios are portable because of the full virtualization strategy adopted by NIEP. We also show the feasibility of NIEP in a case study considering a Fog computing and Virtual Customer Premises Equipment (vCPE) scenario. We expect that NIEP will assist network operators in the offline analysis of the functionality and performance of VNF deployments. Pre-tested configurations can be evaluated and optimal configurations may be established before actual VNFs are deployed in the network infrastructure.

The remaining of this paper is organized as follows. In Section II, background and related work are reviewed. In Section III, we introduce NIEP and detail each of its components.

[1]Available at http://ufsm.br/gt-fende

In Section IV, a case study to evaluate the feasibility of the platform is presented. Finally, in Section V, conclusions and comments on future work are presented.

## II. Background and Related Work

In this section, background on NFV and virtualization is presented, highlighting the benefits of NFV over traditional network architectures and discussing the challenges for the successful deployment of NFV-based solutions. In addition, issues related to NFV prototyping are discussed, highlighting the pros and cons of existing frameworks for NFV experimentation.

### *A. Network Functions Virtualization (NFV)*

NFV has been proposed by the European Telecommunications Standards Institute (ETSI) as a novel paradigm to develop, manage, and deploy network services in an easy, cost-effective, and flexible way [8]. In NFV, services commonly performed by dedicated middleboxes are decoupled from hardware and implemented as Virtual Network Functions (VNFs).

VNFs can be deployed on COTS virtualization servers, where each VNF performs a specific function. Multiple VNFs can be deployed on different locations of the infrastructure substrate and chained together in Service Function Chains (SFC) [9]. The benefits of NFV include: *(i)* reduced capital/operational expenditure (CAPEX/OPEX) and energy costs thanks to the use of high-density servers; *(ii)* reduced time to deploy and configure new services; *(iii)* increased flexibility to dynamically migrate and scale (up and down) services; and *(iv)* more concrete potential for new developers to enter the market, since VNFs are typically implemented in software and deployed on available equipment (*e.g.*, whitebox switches). When used with other technologies, such as Software-Defined Networking (SDN) [10], the substrate network can be more easily customized to fulfill specific needs of customers.

NFV also enables the use of Service Function Chains [9], in which multiple VNFs, each with their own specific purpose, can be chained together in order to provide complex network services. Since those VNFs are independent from each other, they can be deployed on different virtualization servers. Figure 1 shows an example of an SFC, where three servers are employed, each running their own hypervisor and set of VNFs, and with lines representing different SFCs.

Despite its benefits, NFV increases the complexity and implies major changes to the network infrastructure every time new VNFs are deployed. Therefore, new toolsets for VNF emulation, evaluation, and debugging are needed. In traditional networks, operators can use tools such as *tcpdump*, *ping*, and *traceroute* to identify problems in the network, while bugs on middleboxes should be corrected by providers. In NFV, while those tools can still be useful, new network components must be monitored and analyzed to identify problems such as bottlenecks, failures, misconfiguration, or implementation bugs. For SDN, Mininet [11] uses virtualization technologies to deploy and test virtual networks, which can be used with an external SDN controller for experimentation purposes. However, Mininet does not offer support for experimentation with VNFs because of its focus on SDN. As a consequence, new frameworks have been proposed where Mininet is integrated with other software components that can be used to deploy and manage VNFs. In the next subsection, we present and discuss existing frameworks for NFV experimentation.

### *B. NFV Experimentation Frameworks*

Different NFV experimentation frameworks have recently emerged. Among them, EsCAPE [12] has received attention from the NFV community, figuring as a prototyping framework from the UNIFY architecture, consisting of three abstraction layers: Service Layer, Orchestrator Layer, and Infrastructure Layer. EsCAPE provides a common platform that enables users to prototype and orchestrate SFCs whose VNFs are deployed as containers running Click [13], as well as a built-in VNF catalog with basic network functions. EsCAPE's network infrastructure consists of a Mininet instance with OpenVSwitches [14] connected to an external SDN controller (POX) responsible for steering traffic between VNFs. It also supports development and test of orchestration components, extending Mininet to work with NETCONF. The focus of EsCAPE is thus on the creation and management of SFCs, although it can be used to prototype other topologies as well.

MeDICINE [15] is an NFV prototyping platform able to emulate a multi-PoP environment with production-ready network functions running on containers. MeDICINE is based on a previous work called ContainerNET[2], which extends the Mininet framework to support container-based VNFs. Using the Mininet API, links between complex multi-PoP environments are established taking into account requirements in terms of delay, bandwidth, and packet loss rate. Docker[3] is used in MeDICINE to deploy VNFs on these PoPs. MeDICINE also provides end-points for each emulated PoP, enabling users to connect their emulated topology to existing MANO tools.

Both EsCAPE and MeDICINE use containers for deploying and executing VNFs. Although this is enough for most NFV use cases [16], container-based virtualization presents issues for a number of specific NFV scenarios. For example, in comparison to hypervisor-based virtualization, containers do not provide cross-platform compatibility and their life-cycle management is onerous [17]. Moreover, as opposed to Virtual Machines (VMs), containers increase the attack surface of a host [18], since each OS image has its own set of vulnerabilities and share the same kernel. In scenarios with heterogeneous networks, different servers with different operating systems form the infrastructure substrate, such as vCPE, virtual Evolved Packet Core (vEPC) and Fog Computing. In this case, the same VNF can be deployed and migrated to any point of the infrastructure without code changes. Security is also a major concern, making containers not well-suited for prototyping these scenarios.

[2]Available at https://github.com/containernet/containernet
[3]Available at http://www.docker.org/

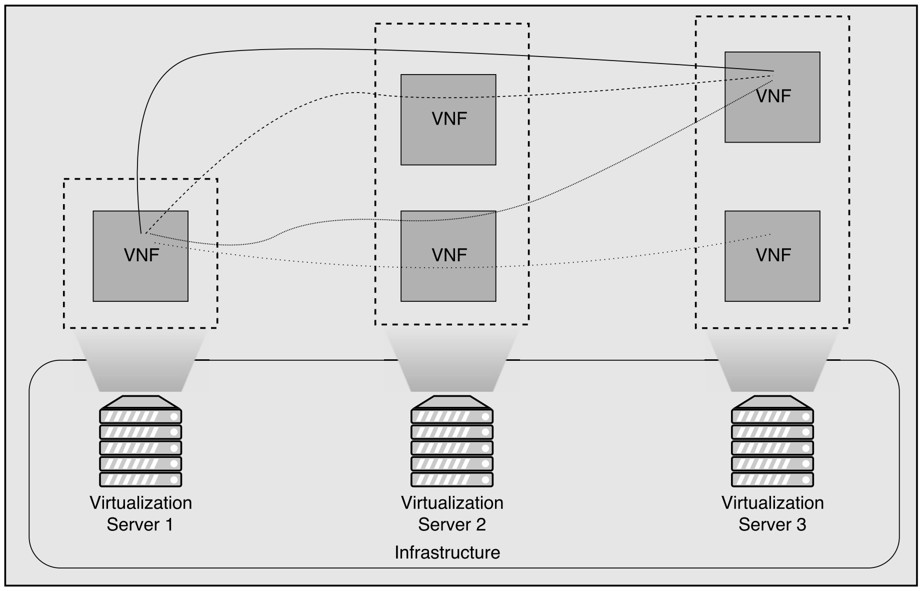

Fig. 1. Example of a Service Function Chain

Other frameworks like MaxiNet [19] present a distributed way to deploy SDN topologies over multiple physical machines. Although it does offer support to the deployment of containers, MaxiNet does not provide a way of using these containers to host VNFs. It is important to note that our solution is the first to provide a prototyping framework with focus on the emulation of different NFV scenarios.

Considering the lack of experimental frameworks for heterogeneous NFV scenarios, the focus of existing platforms on SFCs and multi-pop environments, and the possible security flaws inherent to container-based virtualization, we introduce in the next section NIEP, a framework that integrates a minimal VNF platform (Click-on-OSv) with Mininet, allowing prototyping and evaluating diverse scenarios with hypervisor-based VNFs, which can then be subsequently deployed on production environments.

## III. NIEP: NFV Infrastructure Emulation Platform

In this section, NIEP is described in details. First, in Section III-A we discuss a set of requirements identified that must be satisfied by the proposed platform. Next, in Section III-B the modules of NIEP are presented, detailing their operation. Finally, in Section III-C, details about the interactions between the modules are provided, describing the operation flow of NIEP.

### A. Simulation Requirements

Emulation plays an important role in the design, development, and analysis of VNFs, especially for innovative functions and services. The emulation of a real system should represent it as accurately as possible, defining a synthetic environment and submitting it to real live testings [20]. The use of emulated environments has increased significantly in recent years, since they enable the evaluation of large-scale systems at reduced costs, before actual deployment. However, the development of emulation platforms, in particular for NFV scenarios, must satisfy a set of fundamental requirements, asand accurately represent real network environments. Aiming to create a novel platform for the evaluation of NFV-based infrastructures, we took into consideration a set of requirements previously identified by Varga and Horing [21], Baumgat, Heep and Krause [22], and Schaeffer-Filho *et al.* [23]. These requirements are summarized as follows:

- **Scalability**: the platform must be able to perform simulations with a large number of nodes;
- **Flexibility**: the platform should facilitate the emulation process, as the user must be able to easily specify network topologies. Also, topology elements (*e.g.,* hosts, switches, VNFs) must be generic enough to be reused in a range of scenario definitions;
- **Remodeling**: the definition of the network scenario must be simple, dynamic, and fast for prototyping, thus allowing the network topology to be easily modified as needed;
- **Software Execution**: the elements provided must be similar to real components existing in production network environments, providing reliable experimental results.

### B. NIEP Architecture

NIEP is based on the integration of existing tools for VNF design (Click-on-OSv), VM management (KVM hypervisor), and network emulation (Mininet), with an orchestration module, which is the core element of NIEP. The overall view of the NIEP architecture, comprising the aforementioned tools as well as the orchestrator, is presented in Fig. 2.

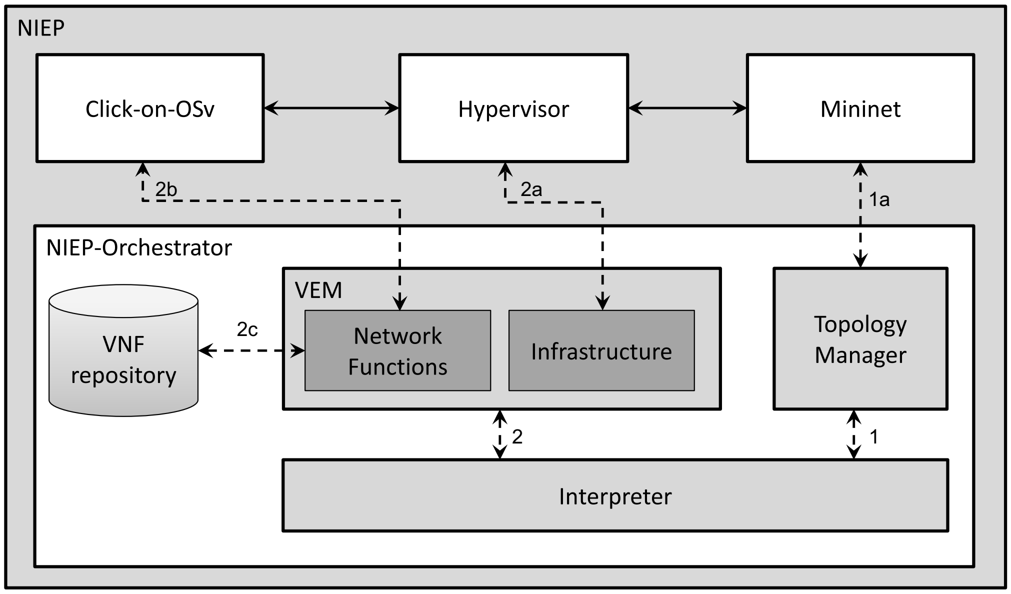

Fig. 2. NIEP Architecture

The first module of NIEP is Click-on-OSv [6], an OSv-based operating system specially built for NFV experimentation. Click-on-OSv leverages on the Click Modular Router [24] to create and execute network functions, and provides a Representational State Transfer (REST) interface for controlling the underlying operations (*e.g.,*., metrics monitoring and life-cycle management). Since Click-on-OSv is a complete virtual machine, it simplifies the controlling and provisioning processes due to its independence from the host operational system. Moreover, it is possible to remotely create VMs using a set of heterogeneous servers, sharing resources (*e.g.,* memory, processing and network), and performing VNF functions in a distributed way.

NIEP uses a KVM hypervisor, which is a virtual VM manager that implements full virtualization, to support the execution of multiple VMs running images of different types of operating systems. The Virsh tool[4] is used by the NIEP orchestrator to manage the KVM virtual machines. It is a Command Line Interface (CLI) that enables controlling VMs through system calls. Also, KVM provides better throughput rates for the Click-on-OSv platform due to the VirtIO optimized implementation[5]. These virtualization upgrades make the packet processing of Click running on OSv faster than other hypervisors (*e.g.* VirtualBox, Xen).

Mininet [11] is a widely used network emulator that relies on process-level virtualization. This type of lightweight virtualization emulates guest machines as isolated processes, reserving memory, CPU and network, inheriting the host functions and programs, and enabling the design of large-scale network environments. In NIEP, Mininet hosts are used to represent servers and client machines, OpenFlow switches and controllers are deployed to enable SDN technology, and links are specified to virtually connect the described elements.

Network topologies in the NIEP platform can be defined using a JavaScript Object Notation (JSON) file, which simplifies the infrastructure deployment process when compared to Mininet. Thus, users can configure a Mininet topology with hosts, switches, and controllers while defining other useful information such as resource allocation for VNFs and connections among them (*e.g.,* the capability of creating SFCs).

The NIEP-Orchestrator is responsible for the platform end user interfacing. It receives the JSON topology definition and executes all the necessary actions to perform the instantiation process. This module comprises four elements, each one providing a set of methods to execute specific actions that jointly control the NIEP environment:

- **VNF Repository**: responsible for storing the virtualized network functions, which are implemented as Click scripts. Since the defined VNFs can be distributed in many machines, the repository must be widely accessible for all VEM instances. Thus, technologies such as Hadoop Distributed File System (HDFS) or common marketplaces can be used as the VNF repository module;
- **Virtualized Elements Manager (VEM)**: responsible for controlling the execution of VNFs and providing the communication interfaces (*e.g.,*., network bridges). This element is composed of two functional blocks, the Network Functions, which directly controls the Click-on-OSv instances through a REST interface, and the Infrastructure, that controls the KVM hypervisor execution using the Virsh CLI;
- **Topology Manager**: responsible for creating and initial-

[4]https://libvirt.org/
[5]https://www.linux-kvm.org/page/Virtio

izing the Mininet network topology on the platform. It creates all the requested elements (*e.g.,*., hosts, switches, controllers) through the Mininet API and saves them for future user operations;
- **Interpreter**: responsible for validating the JSON topology definition and handling user requests (*e.g.,*., creating a new network topology or retrieving statistical data). This module works as data input, receiving NIEP topologies, and output, returning emulation and topology modifications requests results.

### C. Module Interactions

The modules of the NIEP architecture presented in the previous section are integrated by the Orchestrator, coordinating the instantiation of Mininet network topologies and VNF emulation. Initially, the Orchestrator module, upon receiving a JSON topology specification, forwards it to the Interpreter element which proceeds with its validation, essentially analyzing the presence of mandatory elements and the correctness of the configuration. The Interpreter then separates the available information into two sets: one related to the Mininet network topology definition (*e.g.,* hosts, switches, controllers) and another related to the VNFs to be executed (*e.g.,* memory, CPU, interfaces, Click network function).

The first set of information computed by the Interpreter module is sent to the Topology Manager – indicated by (1) in Fig. 2, which receives the data about the definition of the Mininet environment. The Topology Manager, after processing the information to create the requested topology, initializes it in the Mininet emulator – (1a) in Fig. 2.

The VEM element receives the second set of information sent by the Interpreter (2). It verifies the actions to be performed, and forwards them to the Network Functions and Infrastructure functional blocks. The first to execute is the Infrastructure block (2a), which creates virtual machines using the Hypervisor and the communication links to the network topology in Mininet, through a bridge interface. At the end of this process, the Functions Virtualization block (2b) starts the Click-on-Osv platform by fetching the user-defined Click function from the VNF Repository (2c).

The integrated platform described in this section allows the rapid prototyping and evaluation of large-scale NFV scenarios. We believe this can be a valuable toolset in the repertoire of network operators, which will be able to assess the functionality and evaluate the performance of individual VNFs as well as SFCs before their actual deployment in production networks.

## IV. Case Study and Experimental Evaluation

In this section, an experimental evaluation to demonstrate the effectiveness of the proposed solution is presented. First, in Section IV-A, the case study scenario used in our evaluation is detailed. Next, in Section IV-B, the results obtained are presented and discussed. Finally, qualitative metrics are discussed in Section IV-C. The main objective of the performed experiments is to provide a benchmark of our platform on emulating heterogeneous scenarios with topologies of different sizes, as well as to assess how NIEP meets the requirements defined in Section III.

### A. Case Study Scenario

The experimental setup defined is composed of two locations: the Customer Premises (CP) and an Internet Service Provider (ISP), as shown in Fig. 3. In CP, a Mininet host acting as a client is connected to a VNF with limited resources (1 core, 192MB RAM) running a static router to emulate Customer Premises Equipment (CPE) connected to an ISP. At the ISP side, a VNF with more resources (2 cores, 2GB RAM) running a firewall is connected to a Mininet topology with a virtual OpenFlow switch (OpenVSwitch), which in turn is connected to an SDN Controller and a host acting as an application server.

Four different configurations were used to evaluate how the number of customers connected to the ISP impact the performance of NIEP, the number of CPEs as 2, 4, 8 and 16. In addition, a second setup was devised, in which two firewall VNFs were deployed on the ISP side as a way to balance the load imposed by the CPEs.

As for the physical infrastructure used, all CP instances were deployed on an Intel Core i7-6700k@4.00GHz server, with 8GB RAM DDR4, 4 cores, and running CentOS 7. In turn, the ISP was deployed on a server with Intel Xeon E3-1220v6@3.00GHz, 8GB RAM DDR4, 4 cores, running Ubuntu 14.04. Servers were connected by a 1Gbps physical link, in which the CPEs share the available bandwidth. To deploy the Mininet VM and VNFs, the KVM hypervisor was used in both servers.

### B. Results

We performed 30 executions for each setup and number of CPs to reach a confidence interval of 99% on all experiments. NIEP can be used to evaluate diverse NFV scenarios, providing a fine-grained control over several configuration parameters. The defined topology can be easily changed, for example, to test different network paths, adding, removing, and reconfiguring hosts, or changing link properties. As such, we consider that boot time is an important metric to be evaluated, since it can impact the process of changing and evaluating these topologies.

To evaluate NIEP, its components were instrumented to report their boot time to a centralized server. In the defined scenario, the longest boot time corresponds to the VNFs on the CP side, because their number increases across the tests, putting more stress on the physical server, while the number of VNFs on the server emulating the ISP changes only between setups. All VNFs are initialized in parallel, but only after Mininet. Due to this, the measured boot time is the time Mininet takes to initialize plus the average boot time of the VNFs on the CP side, in order to represent the time it takes for the entire topology to be ready. The obtained results are summarized in Fig. 4.

In the second setup, two firewall instances were deployed instead of one, so the boot time is slightly higher on the CPs

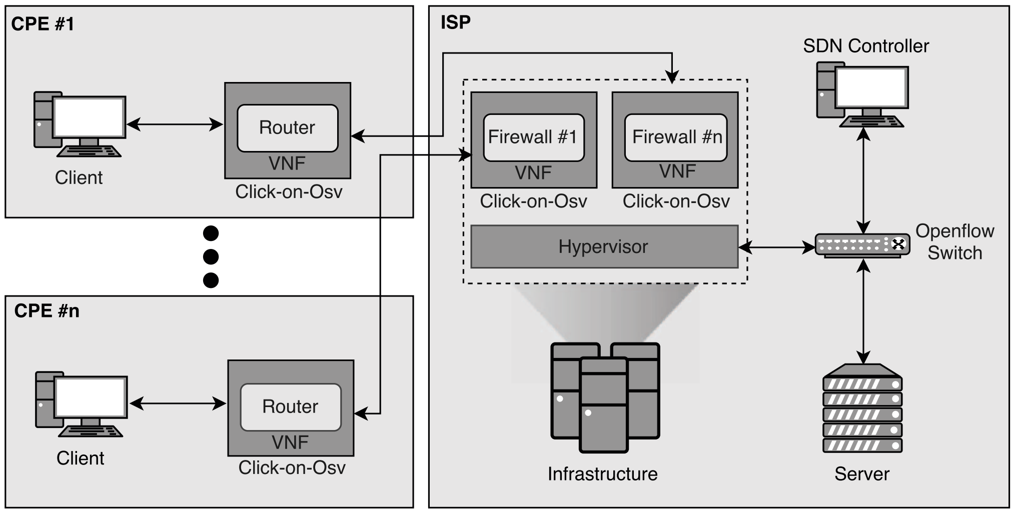

Fig. 3. Experimentation Scenario.

VNFs, as they need additional configuration to send traffic to different firewalls.

The exception to this is when deploying two CP instances (first two bars in Fig. 3), where setup 1 takes 529 ms with an error margin of 3 ms, while setup 2 takes 532 ms with an error margin of 2.8 ms. In particular, this occurs because with only two CP instances there are still free physical cores left to be used exclusively by the hypervisor and operating system on tasks related to configuring and deploying virtual machines.

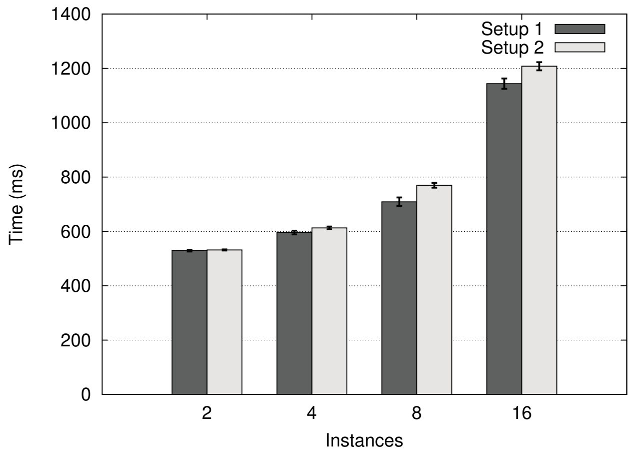

Fig. 4. NIEP's average boot time.

To evaluate NIEP performance under heavy load, the throughput can be analyzed to identify unintended bottlenecks caused by external factors to the emulation (*e.g.,* CPU and memory limitations). For example, if physical resources were strangled, the limited ability to process packets would be reflected in a throughput between CPs and the ISP smaller than the link speed (1Gbps). *iperf* [25] was used to measure NIEP performance, with all CPs sending traffic at the same time to the server running at the ISP side. The values obtained from each CP in each evaluation round were aggregated, since CPs were sharing the same 1Gbps link which connected all instances with the ISP side.

As shown in Fig. 5, in all test cases for both setups the bottleneck was always the link between the CPs and the ISP. This way, increasing the number of instances did not affect the link throughput, meaning that NIEP scaled well on the scenario and setups evaluated.

## C. Discussion

Regarding scalability, NIEP is able to simulate complex scenarios, varying the number of hosts, switches and VNFs, as well as their topology. This behavior is possible because NIEP's VNFs do not run within Mininet. Instead, these VNFs are connected through external bridges, which allow them to be run in remote locations. On the other hand, although EsCAPE can also simulate complex scenarios, the VNFs need to be deployed on the same host, since containers are defined inside Mininet and connections must be established locally.

In contrast to Mininet, topologies can be defined in a higher level by using JSON, which simplifies the infrastructure deployment process. Users can define, in addition to hosts, switches, and controllers, different VNF types, amount of allocated resources for VNFs, and the connections among them, including the capability of creating SFCs.

The use of hypervisor-based virtualization for deploying VNFs enables the emulation of heterogeneous network infrastructures, since a VNF can be directly deployed on any server and operating system running a compatible hypervisor (*e.g.,* KVM, Xen, and VirtualBox) without requiring any change to the VNF source code. This issue is a limiting factor in the platforms discussed in Section II-B, which rely on container-based virtualization.

Finally, containers have inherent security flaws due to the use of namespaces in a shared kernel [26]. These flaws do

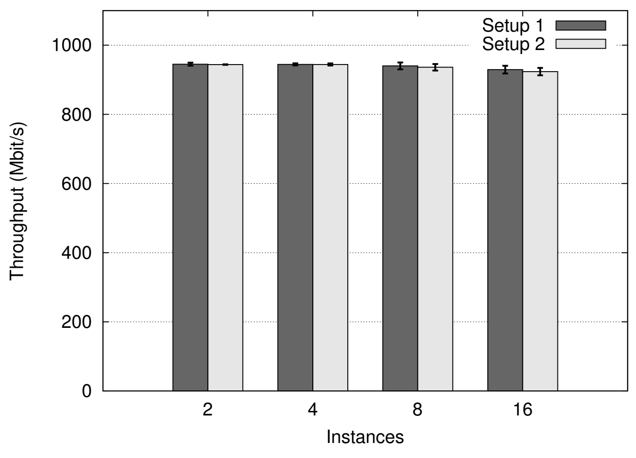

Fig. 5. NIEP's throughput.

not affect actual VMs since they are isolated from the kernel. Therefore, NIEP offers more security when deploying and testing third-party VNFs, and enables the emulation of NFV security scenarios more precisely.

## V. Conclusion and Future Work

The Network Function Virtualization (NFV) paradigm aims to decouple network functions from its associated hardware, replacing them into a software plane by using existing virtualization technologies. However, despite it's advantages, the design and experimentation of Virtualized Network Functions (VNFs) is still a burdensome process. In this way, emulation platforms can be used to execute VNFs in realistic environment, thus contributing to the widely adoption of NFV by network operators.

In this context, this paper presented NIEP, an NFV Infrastructure Emulation Platform based on Click-on-OSv and Mininet. NIEP allows the emulation of different NFV scenarios and VNF design and evaluation, supporting the emulation of heterogeneous infrastructures, such as Fog computing scenarios, composed of devices with different characteristics.

An experimental scenario was designed to evaluate the boot time and throughput of VMs in NIEP, composed of CP and ISP sites. The analysis presented shows that the boot time of VNFs increases almost linearly, indicating almost no impact of NIEP in the instantiation process of VNFs. Moreover, increasing the number of VNFs does not interfere in the throughput achieved by them.

As future work, we aim to provide a user-friendly web user interface for network operators, where information regarding VNFs operation acquired through a REST API can be provided to network operators. Additionally, support to different VNF development technologies can be achieved, adding support to solutions like nginx and BRO for VNF design.

## Acknowledgements

This research was performed partially within the project "Federated Ecosystem for Offering, Distribution, and Execution of Virtual Network Functions" (GT-FENDE). The authors would like to thank Rede Nacional de Ensino e Pesquisa (RNP), for their support to the GT-FENDE project.

## References

[1] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

[2] K. Lu, S. Liu, F. Feisullin, M. Ersue, and Y. Cheng, "Network function virtualization: opportunities and challenges," *IEEE NETWORK*, vol. 29, no. 3, pp. 4–5, May 2015.

[3] J. Garay, J. Matias, J. Unzilla, and E. Jacob, "Service description in the nfv revolution: Trends, challenges and a way forward," *IEEE Communications Magazine*, vol. 54, no. 3, pp. 68–74, March 2016.

[4] M. Imran, A. M. Said, and H. Hasbullah, "A survey of simulators, emulators and testbeds for wireless sensor networks," in *2010 International Symposium on Information Technology*, vol. 2, June 2010, pp. 897–902.

[5] D. Salopek, V. Vasi, M. Zec, M. Mikuc, M. Vaarevi, and V. Konar, "A network testbed for commercial telecommunications product testing," in *2014 22nd International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, Sept 2014, pp. 372–377.

[6] L. da Cruz Marcuzzo, V. F. Garcia, V. Cunha, D. Corujo, J. P. Barraca, R. L. Aguiar, A. E. Schaeffer-Filho, L. Z. Granville, and C. R. dos Santos, "Click-on-osv: A platform for running click-based middleboxes," in *Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on*. IEEE, 2017, pp. 885–886.

[7] 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.

[8] M. Chiosi, S. Wright *et al.*, "Network functions virtualisation (nfv)," *ETSI NFV ISG, White Paper*, vol. 1, 2012.

[9] J. Halpern and C. Pignataro, "Service function chaining (sfc) architecture," Internet Requests for Comments, RFC 7665, 2015.

[10] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Communications Magazine*, vol. 53, no. 2, 2015.

[11] M. Team, "Mininet: An instant virtual network on your laptop (or other pc)," 2012.

[12] B. Sonkoly, J. Czentye, R. Szabo, D. Jocha, J. Elek, S. Sahhaf, W. Tavernier, and F. Risso, "Multi-domain service orchestration over networks and clouds: A unified approach," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, ser. SIGCOMM '15. New York, NY, USA: ACM, 2015, pp. 377–378. [Online]. Available: http://doi.acm.org/10.1145/2785956.2790041

[13] R. Morris, E. Kohler, J. Jannotti, and M. F. Kaashoek, "The click modular router," *ACM SIGOPS Operating Systems Review*, vol. 33, no. 5, pp. 217–231, 1999.

[14] B. Pfaff, J. Pettit, T. Koponen, E. J. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar *et al.*, "The design and implementation of open vswitch." in *NSDI*, 2015, pp. 117–130.

[15] M. Peuster, H. Karl, and S. van Rossem, "Medicine: Rapid prototyping of production-ready network services in multi-pop environments," in *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov 2016, pp. 148–153.

[16] G. NFV, "001: Network functions virtualisation (nfv); use cases, v 1.1. 1," *ETSI, December*, 2013.

[17] R. Morabito, J. Kjllman, and M. Komu, "Hypervisors vs. lightweight virtualization: A performance comparison," in *2015 IEEE International Conference on Cloud Engineering*, March 2015, pp. 386–393.

[18] A. A. Mohallel, J. M. Bass, and A. Dehghantaha, "Experimenting with docker: Linux container and base os attack surfaces," in *2016 International Conference on Information Society (i-Society)*, Oct 2016, pp. 17–21.

[19] P. Wette, M. Draxler, A. Schwabe, F. Wallaschek, M. H. Zahraee, and H. Karl, "Maxinet: Distributed emulation of software-defined networks," in *Networking Conference, 2014 IFIP*. IEEE, 2014, pp. 1–9.

[20] M. Carson and D. Santay, "Nist net: a linux-based network emulation tool," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 3, pp. 111–126, 2003.

[21] A. Varga and R. Hornig, "An overview of the omnet++ simulation environment," in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, p. 60.

[22] I. Baumgart, B. Heep, and S. Krause, "Oversim: A flexible overlay network simulation framework," in *IEEE Global Internet Symposium, 2007*. IEEE, 2007, pp. 79–84.

[23] A. Schaeffer-Filho, A. Mauthe, D. Hutchison, P. Smith, Y. Yu, and M. Fry, "Preset: A toolset for the evaluation of network resilience strategies," in *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*. IEEE, 2013, pp. 202–209.

[24] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The click modular router," *ACM Transactions on Computer Systems (TOCS)*, vol. 18, no. 3, pp. 263–297, 2000.

[25] A. Tirumala, F. Qin, J. Dugan, J. Ferguson, and K. Gibbs, "Iperf: The tcp/udp bandwidth measurement tool," *http://iperf.fr*, 2005.

[26] T. Combe, A. Martin, and R. D. Pietro, "To docker or not to docker: A security perspective," *IEEE Cloud Computing*, vol. 3, no. 5, pp. 54–62, Sept 2016.