On the design and development of emulation platforms for NFV-based infrastructures

Vinícius Fülber Garcia*,
Giovanni Venâncio de Souza
and Elias Procópio Duarte Jr.

Department of Informatics,
Federal University of Paraná,
Curitiba, Paraná, Brazil
Email: vfgarcia@inf.ufpr.br
Email: gvsouza@inf.ufpr.br
Email: elias@inf.ufpr.br
*Corresponding author

Thales Nicolai Tavares,
Leonardo da Cruz Marcuzzo
and Carlos R.P. dos Santos

Department of Applied Computing,
Federal University of Santa Maria,
Santa Maria, Rio Grande do Sul, Brazil
Email: tntavares@inf.ufsm.br
Email: lmarcuzzo@inf.ufsm.br
Email: csantos@inf.ufsm.br

Muriel Figueredo Franco,
Lucas Bondan,
Lisandro Zambenedetti Granville
and Alberto Egon Schaeffer-Filho

Federal University of Rio Grande do Sul,
Porto Alegre, Rio Grande do Sul, Brazil
Email: mffranco@inf.ufrgs.br
Email: lbondan@inf.ufrgs.br
Email: granville@inf.ufrgs.br
Email: alberto@inf.ufrgs.br

Filip De Turck

INTEC,
Ghent University,
Ghent, Belgium
Email: filip.deturck@ugent.be

Abstract: Network Functions Virtualisation (NFV) presents several advantages over traditional network architectures, such as flexibility, security, and reduced CAPEX/OPEX. In traditional middleboxes, network functions are usually executed on specialised hardware (e.g., firewall, DPI). Virtual Network Functions (VNFs) on the other hand, are executed on commodity hardware, employing Software Defined Networking (SDN) technologies (e.g., OpenFlow, P4). Although platforms for prototyping NFV environments have emerged in recent years, they still present limitations that hinder the evaluation of NFV scenarios such as fog computing and heterogeneous networks. In this work, we present NIEP: a platform for designing and testing NFV-based infrastructures and VNFs. NIEP consists of a network emulator and a platform for Click-based VNFs development. NIEP provides a complete NFV emulation environment, allowing network operators to test their solutions in a controlled scenario prior to deployment in production networks.

Keywords: NFV; VNF; emulation; platform; infrastructure; Click; Mininet; network.

Copyright © 2020 Inderscience Enterprises Ltd.

Reference to this paper should be made as follows: Garcia, V.F., de Souza, G.V., Duarte Jr., E.P., Tavares, T.N., da Cruz Marcuzzo, L., dos Santos, C.R.P., Franco, M.F., Bondan, L., Granville, L.Z., Schaeffer-Filho, A.E. and De Turck, F. (2020) 'On the design and development of emulation platforms for NFV-based infrastructures', *Int. J. Grid and Utility Computing*, Vol. 11, No. 2, pp.230–242.

Biographical notes: Vinícius Fülber Garcia is a PhD student in Computer Science at the Department of Informatics of the Federal University of Paraná (UFPR, Brazil) under the supervision of Prof. Dr. Elias Procópio Duarte Jr. He holds a Computer Science degree from Federal University of Santa Maria (UFSM, Brazil) and a Master degree in Computer Science from UFSM Post-Graduate Program in Computer Science. His research interests include, but not limited to, network functions virtualisation and information theory.

Giovanni Venâncio de Souza is a PhD student in Computer Science at the Department of Informatics of the Federal University of Paraná (UFPR, Brazil) under the supervision of Prof. Dr. Elias Procópio Duarte Jr. He holds an MSc (2017) in Computer Science and a Computer Science degree (2016) at the same institution. His research interests include network function virtualisation and fault-tolerant distributed systems.

Elias Procópio Duarte Jr. is a Full Professor at Federal University of Parana, Curitiba, Brazil, where he is the leader of the Computer Networks and Distributed Systems Lab (LaRSis). His research interests include computer networks and distributed systems, their dependability, management, and algorithms. He has published more than 200 peer-reviewed papers and supervised more than 130 students. He is currently Associate Editor of the *IEEE Transactions on Dependable and Secure Computing*, and has served as chair of more than 20 conferences and workshops in his fields. He received a PhD. degree in Computer Science from Tokyo Institute of Technology, Japan, 1997, MSc degree in Telecommunications from the Polytechnical University of Madrid, Spain, 1991, and both BSc and MSc degrees in Computer Science from Federal University of Minas Gerais, Brazil, 1987 and 1991, respectively. He is a member of the Brazilian Computing Society and a Senior Member of the IEEE.

Thales Nicolai Tavares is a graduate of the course on Computer Network Technology at the Federal University of Santa Maria (2016) in Brazil. He is currently a substitute lecturer at the polytechnic school of the same university. He has knowledge in the area of computing, with emphasis on computer networks. His research interests are in network management, software networks and virtualisation of network functions.

Leonardo da Cruz Marcuzzo holds a degree in Computer Science from Federal University of Santa Maria (UFSM) and currently is a MSc candidate in Computer Science at the same institution. His research interests include network functions virtualisation and operating systems.

Carlos R.P. dos Santos is Adjunct Professor of Computer Science at the Department of Applied Computing of the Federal University of Santa Maria (UFSM), Brazil. He holds PhD (2013) and MSc (2008) degrees in Computer Science, both received from the Federal University of Rio Grande do Sul (UFRGS), where he was also Postdoctoral Research Fellow from October 2013 to September 2014. From May 2010 to April 2011 he was a visiting researcher at the IBM T.J. Watson Research Center, Hawthorne, where he developed projects on IT Service Management and Security Management. His current research interests focus on design and management of future networks and technologies, including aspects such as network virtualisation, quality of service management, network programmability, and security management.

Muriel Figueredo Franco is pursuing his PhD under the supervision of Prof. Dr. Burkhard Stiller at the University of Zurich (UZH). He is also a Research Assistant at the Communication Systems Group (CSG). He holds an MSc (2017) in Computer Science from the Federal University of the Rio Grande do Sul (UFRGS) under the supervision of Prof. Dr. Lisandro Granville and obtained a BSc (2014) in Computer Science from the Federal University of Pelotas (UFPEL). His research topics include network functions virtualisation, information visualisation, and blockchain.

Lucas Bondan is a PhD student in Computer Science at the Institute of Informatics of the Federal University of Rio Grande do Sul (UFRGS) in Brazil and an R&D Project Manager at the Brazilian National Research and Educational Network (RNP). From 2016 to 2018 he was a PhD student fellow at the Department of Information Technology of Ghent University in Belgium, working with security-related areas of Network Functions Virtualisation (NFV). He has a Computer Engineering degree from Pontifícia Universidade Católica do Rio Grande do Sul and a Master degree in Computer Science from UFRGS. His research interests include network functions virtualisation, network management and orchestration, service function chaining, cognitive networks, and wireless communication systems.

Lisandro Zambenedetti Granville is Full Professor of Computer Science at the Institute of Informatics of the Federal University of Rio Grande do Sul (UFRGS), Brazil. He holds PhD (2001) and MSc (1998) degrees in Computer Science, both received from UFRGS. From September 2007 to August 2008 he was a visiting researcher at the University of Twente, The Netherlands, with the Design and Analysis of Communication Systems group. He is a member of the Computer Networks Group, where he develops research projects on network and service management. As a Full Professor, he is also involved with supervision and education activities on undergraduate and graduate courses in both Computer Science and Computer Engineering.

Alberto Egon Schaeffer-Filho holds a PhD in Computer Science (Imperial College London, 2009) and is Associate Professor at Federal University of Rio Grande do Sul (UFRGS), Brazil. From 2009 to 2012 he worked as a research associate at Lancaster University, UK. He is a CNPq-Brazil Research Fellow and his areas of expertise are network/service management, network virtualisation and software-defined networks, policy-based management, and security and resilience of networks. He has authored over 60 papers in leading peer-reviewed journals and conferences related to these topics, and also serves as TPC member for important conferences in these areas, including: IFIP/IEEE IM (2019), NetSoft (2019), CNSM (2018), and IEEE/IFIP NOMS (2018). He is the general chair for SBRC 2019, co-chair for IEEE ICC 2018 CQRM Symposium, and demo co-chair for IFIP/IEEE IM 2017.

Filip De Turck is Professor in the Department of Information Technology (Intec) of Ghent University with expertise in network software and research interests in adaptive large-scale data processing and software systems for healthcare, anomaly detection, and resilience of ICT infrastructures and services. In this research area, he is involved in several research projects with industry and academia, serves as Chair of the IEEE Technical Committee on Network Operations and Management (CNOM), chair of the Future Internet Cluster of the European Commission, and is on the TPC of many network and service management conferences and workshops and serves in the editorial board of several network and service management journals. Together with a team of PhD students and postdoctoral researchers, novel techniques and algorithms are designed, and validated by means of large scale evaluation studies, together with partners from industry and academia.

This paper is a revised and expanded version of a paper entitled 'NIEP: NFV Infrastructure Emulation Platform' presented at the 'IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)', 2018, Cracow, Poland.

1 Introduction

Network Functions Virtualisation (NFV) is a novel networking paradigm that fosters innovation and supports the creation of disruptive network services (Mijumbi et al., 2016). In NFV, the network functions are decoupled from the associated hardware and executed in commodity servers (i.e., commercial off-the-shelf servers) by using virtualisation technologies. This shift provides significant advancements in how the networks are designed, maintained, and managed while improving the flexibility, scalability, and cost-benefit of networked environments (Lu et al. ,2015).

All those advantages have brought NFV to the attention of the industry, academia, and standardisation bodies. Several efforts have been conducted for the development of new architectures, systems, and applications for NFV (Garay et al., 2016). Despite a large number of results that have already appeared in this field, several challenges are still open. One of those challenges is to develop a *de facto* approach for predicting the impact of deploying novel Virtualised Network Functions (VNFs) in production environments. In this context, VNF emulation is a promising method that can support the design and evaluation of NFV-based scenarios.

Emulation has proved to be an effective method to evaluate network-based environments, systems and applications (Imran et al., 2010; Salopek et al., 2014). In the same way, providing comprehensive emulation tools that support the specific NFV elements (e.g., Virtualised Infrastructure Manager (VIM) and VNF Manager (VNFM)) is of paramount importance for network operators, researchers, and developers. However, despite its inherent benefits, solutions for NFV emulation are still scarce, limited (e.g., due to low portability or lack of support for heterogeneous environments), usually they are not intuitive, and involve a steep learning curve before they can be fully adopted.

In this paper, we present the NFV Infrastructure Emulation Platform (NIEP)[1], a novel platform based on Click-on-OSv (Marcuzzo et al., 2017) and Mininet (Lantz et al., 2010) that allows VNF evaluation by the emulation of diverse NFV scenarios. NIEP allows operators to rapidly create heterogeneous NFV emulated scenarios. These scenarios are portable because of the full virtualisation 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 effectively 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 rest of this paper is organised as follows. In Section 2, the background and related work are reviewed. We then present the simulation/emulation requirements and the proposed NIEP architecture in Section 3. In Section 4, we discuss the data model employed to specify the network topologies. In Section 5, we describe a case study to demonstrate the feasibility of the platform. Finally, the conclusions follow in Section 6, along with a discussion of future work directions.

2 Background and related work

In this section, we present an overview of NFV and network virtualisation technologies. After reading this section, it should be clear that NFV brings multiple advantages in comparison with traditional network architectures that are based on middleboxes often deployed on specialised hardware. However, it should be also clear that there are challenges for the successful deployment of NFV-based solutions in production networks. This section also presents issues related to NFV prototyping and evaluation are discussed, highlighting the pros and cons of existing frameworks.

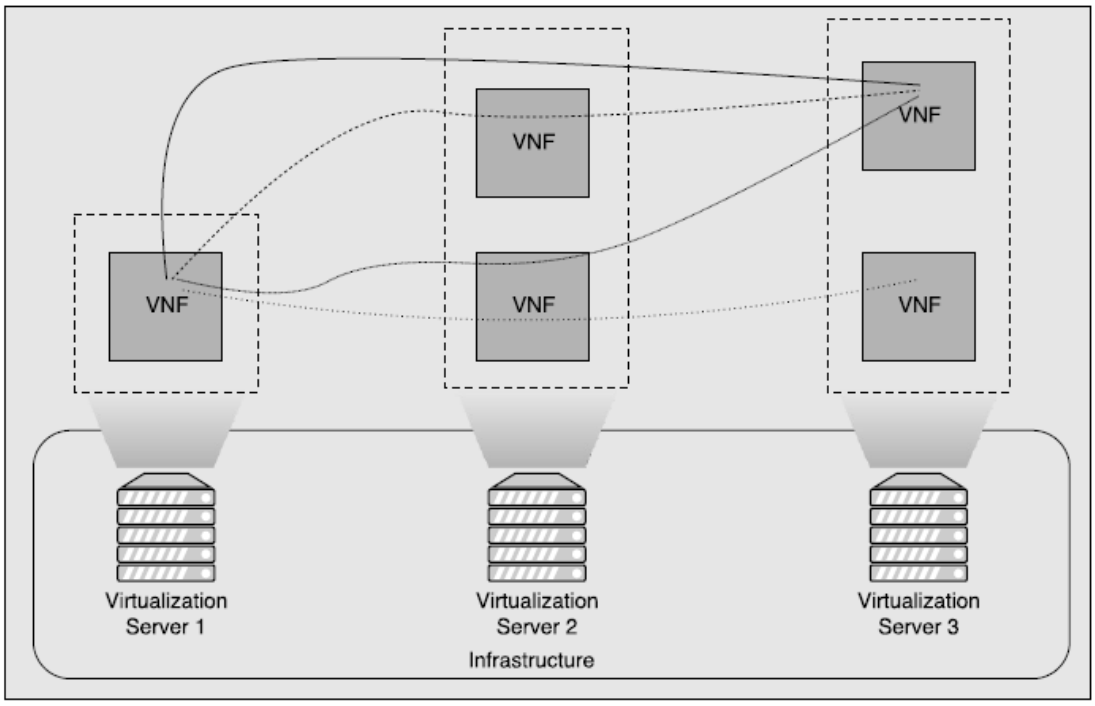
2.1 Network functions virtualisation (NFV)

NFV technology was first proposed and standardised by the European Telecommunications Standards Institute (ETSI) as a paradigm that decouples network functions from dedicated hardware allowing their implementation using virtualisation technology that can be executed on Commercial Off-The-Shelf (COTS) hardware. The fact that NFV does not require specialised hardware and is deployed as a virtual infrastructure enables the development and management of network function in an easy, cost-effective, and flexible way (Chiosi and Wright, 2012). Furthermore, NFV allows the fast creation of new network functions that can be combined to provide complex network services. NFV together with other technologies based on virtualisation has solved the Network Ossification phenomenon (Handley, 2006).

The multiple advantages of NFV technology include: (i) NFV is cheap, in particular in terms of capital/operational expenditures (CAPEX/OPEX) as general purpose hardware can be used; (ii) NFV is fast to deploy, configure and update; (iii) NFV is flexible, as virtual functions can dynamically migrate and by using elasticity technology they can be scaled up and down according to the demand; and (iv) NFV opens up the market, allowing new players to develop for the computer networks market. NFV is often employed with other technologies, such as Software-Defined Networking (SDN) (Han et al., 2015), which allows the substrate network to be more easily customised to fulfill the specific needs of customers.

Individual VNFs may be combined to execute complex network services. Service Function Chains (SFC) (Halpern and Pignataro, 2015) are composed of multiple and independent VNFs that can be executed on different virtualisation environments. The IETF envisions even multi-domain SFCs, which should employ a hierarchy of orchestrators (Bernardos et al., 2018). Figure 1 shows an SFC example with three servers running each a hypervisor and set of VNFs which are interconnected forming multiple SFCs.

Figure 1 A SFC example



Despite the multiple advantages, NFV increases the complexity and it is undeniable that before it is deployed major changes are required to the existing network infrastructures. Therefore, new technologies and tools for VNF evaluation are needed. Although tools such as *tcpdump*, *ping*, and *traceroute* can still be used to identify problems in virtualised networks, in NFV, while those tools can still be useful, new network components must be monitored and evaluated to identify problems such as bottlenecks, failures, misconfiguration, or implementation bugs.

Mininet (Mininet Team, 2012) is a simple yet powerful tool that allows the evaluation of Software Defined Network technology. Mininet can be used with an external SDN controller for running experiments. However, Mininet does not offer support for experimentation with VNFs. As a consequence, new tools have been proposed that integrate Mininet with other software components that can be used to deploy and manage VNFs. In the next subsection, we present and discuss some of those efforts that have been proposed for NFV experimentation.

2.2 NFV experimentation frameworks

A number of tools and frameworks have been recently proposed for the experimental evaluation of NFV technology. EsCAPE, MeDICINE, SONATA, and Maxinet are among some of the most important of those tools and frameworks. These four platforms are described next.

- *EsCAPE* (Sonkoly et al., 2015): Extensible Service Chain Prototyping Environment (EsCAPE) is a prototyping framework developed in the context of 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 (Morris et al., 1999). EsCAPE also features a built-in VNF catalog with basic virtual functions. EsCAPE's network infrastructure is based on Mininet with OpenVSwitches (Pfaff et al., 2015) connected to an external SDN controller (POX) responsible for steering traffic between VNFs. ExCAPE also supports the 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 and evaluate other technologies as well.
- *MeDICINE* (Peuster et al., 2016): The Multi-Datacentre service Chain Emulator (MeDICINE) is an NFV prototyping platform that was designed to emulate multi-PoP environments in which virtual functions are executed on containers. MeDICINE is based on ContainerNET[2], which extends the Mininet framework to support container-based VNFs. Links between complex multi-PoP environments are established using the Mininet API, allowing the specification of multiple requirements such as delay, bandwidth, and the packet loss rate. Docker[3] is used in MeDICINE to deploy VNFs on these PoPs. MeDICINE also provides end-points for each PoP, enabling the interconnection of the elements also to other elements of the ETSI architecture.
- *SONATA* (Karl et al., 2016): SONATA is a tool for NFV composition, testing, and orchestration. It contains an emulation platform based on ContainerNet (Peuster et al., 2016) which allows developers to prototype network services in end-to-end multi-PoP scenarios. The platform also provides APIs for integration with other components and systems based on the ETSI specifications.
- *Maxinet* (Wette et al., 2014): Maxinet is an extension of Mininet that can be executed in a distributed fashion, and in this way supports the emulation of networks. Maxinet works as an abstraction layer connecting multiple Mininet instances running on distinct hosts connected on a network.

EsCAPE, MeDICINE and SONATA use containers for deploying and executing VNFs. Although container technology should be enough for most NFV use cases (GS NFV, 2013), container-based virtualisation presents some issues for specific NFV scenarios. For example, in comparison to hypervisor-based virtualisation, containers do not provide multi-platform compatibility and their life-cycle management is certainly more expensive (Morabito et al., 2015). Moreover, as opposed to Virtual Machines (VMs), containers increase the vulnerability in terms of security threats (Mohallel et al., 2016), since each operating system image has its own set of vulnerabilities and share the same kernel. In scenarios with heterogeneous networks, multiple hosts with different operating systems form the infrastructure substrate, such as vCPE, virtual Evolved Packet Core (vEPC) and Fog Computing. Any given VNF can be deployed and migrated anywhere in the infrastructure.

As for Maxinet, although it also supports virtual nodes in the same way that Mininet does, not all virtualisation technology is available. The main purpose is to run Mininet instances in a distributed way.

In the next section, we introduce NIEP, a framework that integrates a minimal VNF platform (Click-on-OSv) with Mininet, allowing diverse NFV scenarios to be prototyped and evaluated. NIEP fills a gap in terms of the lack of experimental frameworks for evaluating heterogeneous NFV scenarios, as the focus of existing platforms is on SFCs and multi-pop environments. We believe our solution is the first to provide a prototyping framework for the emulation of NFV technology in a variety of scenarios.

3 NIEP: NFV infrastructure emulation platform

In this section, we describe NIEP by first, in Sub-section 3.1 discussing a set of requirements identified that must be satisfied by the proposed platform. Next, in Sub-section 3.2 the NIEP architecture is presented, with a description of all modules of which it is composed. Next, in Sub-section 3.3, the interactions between the modules are described.

3.1 Emulation requirements

Emulation plays an important role in the design, development, and analysis of VNFs, especially for innovative functions and services. The emulation of a system should represent the system as accurately as possible, allowing the execution of real live tests on the emulation environment (Carson and Santay, 2003). The use of these environments has increased significantly in recent years, as they are so convenient for the evaluation of large-scale systems, allowing deep analysis of the system under realistic conditions before it is actually deployed.

An emulation platform for NFV technology should satisfy a set of fundamental requirements. We identified the following requirements raised by Varga and Horing (2008), (Baumgart et al., 2007), and (Schaeffer-Filho et al., 2013) as the basis on which a novel platform for the emulation of NFV-based infrastructures should be designed.

- *Scalability*: the platform must be able to involve a large number of nodes when emulating the NFV-based system;
- *Flexibility*: it should be straightforward to define and update the emulation process, and the user should have a choice of network topologies to specify and use. The building blocks (e.g., hosts, switches, VNFs) with which the system is specified must be generic enough to be reused in a range of scenario definitions;
- *Remodelling*: the definition of evaluation scenarios must be simple, dynamic, allowing fast prototyping; the network topology must be easily modified as needed;
- *Software execution*: the building blocks provided must reflect those of the corresponding actual system in production, thus providing reliable experimental results.

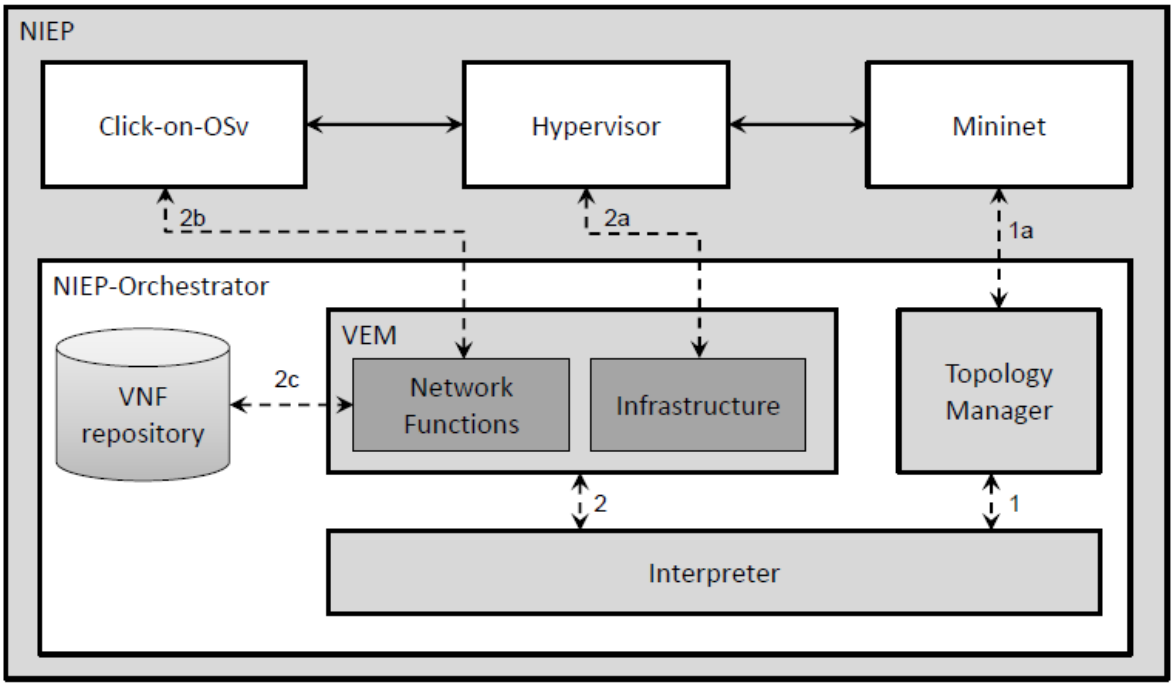
3.2 NIEP architecture

NIEP is based on the integration of existing tools for VNF design (Click-on-OSv), VM management (KVM hypervisor), and network emulation (Mininet), plus a core element, which is the orchestration module. The architecture is shown in Figure 2.

We start the description of the NIEP modules with Click-on-OSv (Marcuzzo et al., 2017), an NFV system based on the single-process operating system OSv. Click-on-OSv leverages the Click Modular Router (Kohler et al., 2000) to create and execute virtual functions and provides a Representational State Transfer (REST) interface for controlling the underlying operations (e.g., monitoring and lifecycle management). Click-on-OSv itself is a complete virtual machine, it simplifies the control and provisioning processes due to its independence from the host operating system. Moreover, it is possible to remotely create VMs on a set of heterogeneous hosts that run VNF functions in a distributed way.

NIEP is based on a KVM hypervisor, which is a virtual VM manager that implements full virtualisation, to support the execution of multiple VMs running images of different 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 VM control with system calls. We highlight that KVM provides better performance for Click-on-OSv due to VirtIO[5]. These virtualisation optimisations make packet processing by Click running on OSv faster than other hypervisors (e.g., VirtualBox, Xen).

Figure 2 NIEP: the architecture



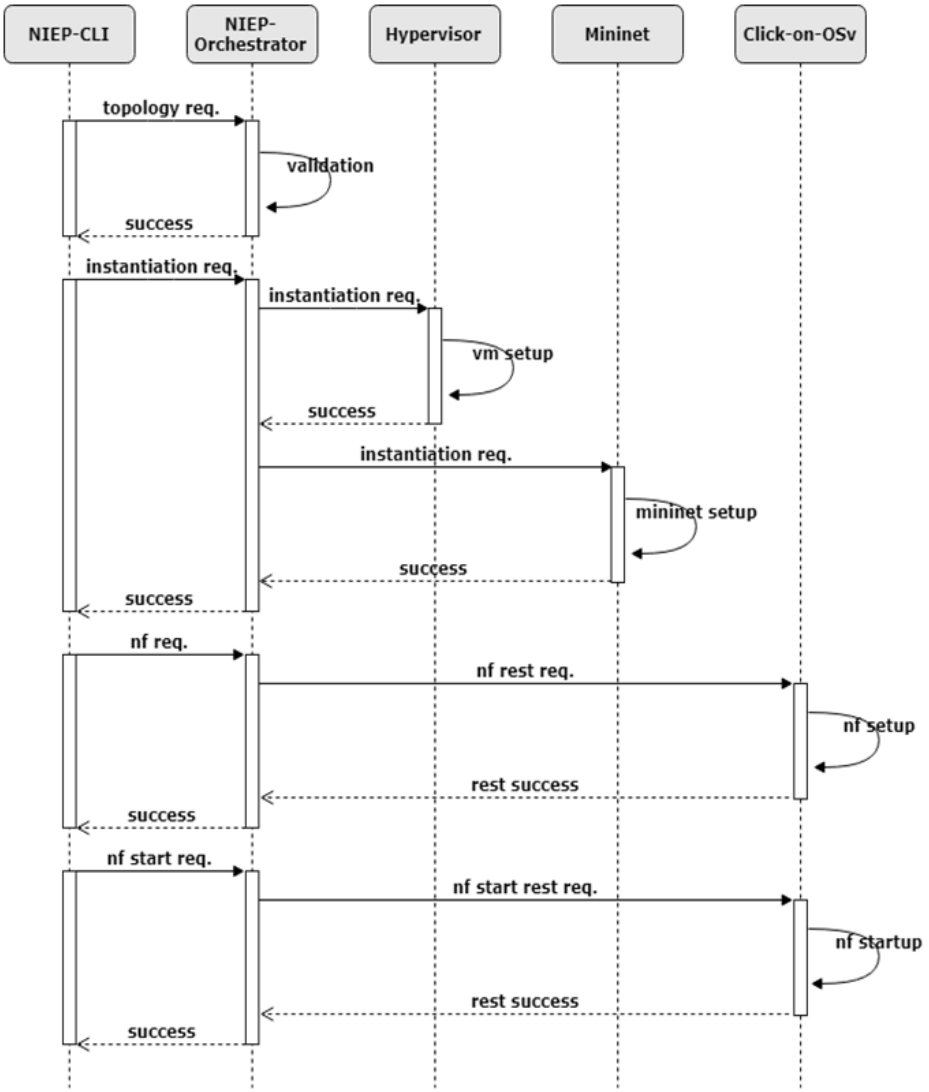
Mininet (Mininet Team, 2012), as mentioned above, is a widely used network emulator that relies on process-level virtualisation. This lightweight virtualisation strategy is used to emulate guest machines as isolated processes, with the proper share of memory, CPU and network resources, enabling the simulation of large-scale network environments. In NIEP, Mininet hosts are used to representing servers and clients, OpenFlow switches and controllers.

Network topologies in NIEP are specified with JSON (JavaScript Object Notation), which also simplifies the infrastructure deployment process when compared to Mininet. Thus, users can configure a Mininet topology with hosts, switches, and controllers also defining other useful information such as resource allocation for VNFs and their interconnections forming SFCs.

The NIEP-Orchestrator provides the user interface. The topology is entered as input, and all required actions are executed to instantiate system. The NIEP-Orchestrator consists of four elements, described next.

- *VNF repository*: this module is responsible for storing VNFs, which are implemented as Click scripts. As VNFs can be deployed in a distributed fashion across multiple hosts, the repository must be universally accessible. Therefore it works as a marketplace where users can share, publish and obtain VNFs;
- *Virtualised elements manager (VEM)*: this module both controls the execution of VNFs and provides communication interfaces (e.g., network bridges). The VEM is composed of two functional blocks, the Network Functions blocks, which directly controls Click-on-OSv instances using a REST interface, and the Infrastructure, that controls the KVM hypervisor execution using the Virsh CLI;
- *Topology manager*: this module allows the creation and initialisation of the Mininet topology. It creates all the specified elements (e.g., hosts, switches, controllers) through the Mininet API which later run user operations;
- *Interpreter*: this component is responsible for validating the topology specifications and handling user requests (e.g., specifying a new network topology or obtaining monitoring data). The input consists of NIEP topology specification, and the output consists of request results.

Figure 3 NIEP topology instantiation



3.3 Module interactions

The modules of the NIEP architecture presented in the previous subsection are integrated by the Orchestrator, which as the name implies orchestrates the VNF emulation on the specified Mininet network topology. Initially, the Orchestrator receives the topology specification, forwards the topology to the Interpreter module which does the required validation, checking for mandatory elements and evaluating the configuration correctness. The Interpreter then organises the information in two sets: one with information on the Mininet network topology (e.g., hosts, switches, controllers) and the other set with information related to the VNFs to be executed (e.g., memory, CPU, interfaces, plus the Click network function itself).

The first information set of computed by the Interpreter is sent to the Topology Manager – labelled with (1) in Figure 2, that processes the data of the Mininet environment. The Topology Manager, after processing the information to create the requested topology, triggers the Mininet emulator (1a). The VEM element receives the second set of information from by the Interpreter (2). It checks actions to be executed, which are forwarded to the Network Functions and Infrastructure blocks. The Infrastructure block (2a) executes first, creating virtual machines using the Hypervisor and the communication links to the network topology in Mininet using a bridge interface.

At the end of this process, the user can make requests to the Functions Virtualisation block (2b) to start Click-on-Osv by fetching the user-defined Click function from the VNF Repository (2c). This process can also be used to deploy a new Click function on the same system instance, by re-uploading and restarting the Click-on-OSv service without having to restart the VM or the entire topology. Figure 3 shows the high-level communication steps to run a NIEP topology.

In synthesis, NIEP is a platform that allows rapid prototyping and evaluation of large-scale NFV scenarios. In this way, it can be a valuable tool for network operators, by allowing the assessment of the functionality and performance of individual VNFs as well as SFCs before their actual deployment in production networks.

4 NIEP data model

In this section, we present the NIEP data model for defining network topologies. The data model includes definitions of VNFs and SFCs and the network topology, including hosts, switches, and controllers. The elements are described in separate JSON files and are described next.

A VNF is described as a JSON object with five properties, as shown in Figure 4. The unique ID is the key to identify a VNF instance and is used along the execution of the emulation execution by the orchestrator to access the required VNFs as it executes tasks including monitoring, deployment, and VNF lifecycle operations. The other properties are used for VM configuration: memory requirements (MEMORY), number of Virtual CPUs (VCPU), and network interfaces (MANAGEMENT_MAC and INTERFACES). The MANAGEMENT_MAC corresponds to a dedicated interface that is employed for the sole purpose of sending and receiving data from the NIEP-Orchestration. The INTERFACES property is used to connect the emulated VNFs and hosts and contains the MAC address and the virtual connection (i.e., network bridge) ID from which data is received.

Figure 4 VNF simplified JSON schema

```
{
  "type": "object",
  "properties": {
    "ID": {"type": "string"},
    "MEMORY": {"type": "integer"},
    "VCPU": {"type": "integer"},
    "MANAGEMENT_MAC": {"type": "string"},
    "INTERFACES": {"type": "array",
      "items": {
        "type": "object",
        "properties": {
          "MAC": {"type": "string"},
          "ID": {"type": "string"}
        }
      }
    }
  }
}
```

SFCs are also specified with a JSON file, Figure 5, composed of five attributes that represent the Service Function Chain. The attributes are as follows. The ID is unique and is used to identify the SFC as a whole, thus making possible to monitor and configure the lifecycle of all the VNFs composing the service. The VNFS attribute represents the set of VNFs that compose the SFC. The VNFS contains the identifier of the VNF in the context of the SFC, as well as a path for the VNF JSON file (created using the schema presented in Figure 4). The VNFs are connected each with a single Incoming Point (IP) and one or more Outgoing Points (OP). The boundary nodes (IP/OP) represent the first (IP) and last (OP) point of a service chain. Both IP and OP are represented by an identifier and a virtual connection is used for the connection of the elements of the SFC.

In the specification of a VNF in a SFC Descriptor, the INTERFACE attribute is omitted and the CONNECTIONS attribute is employed instead. The CONNECTIONS attribute consists of four elements: Input Logical Link (ILL), Output Logical Link (OLL), and the associated MAC addresses (when necessary). The network traffic is delivered to a VNF from an ILL and, after being processed, the traffic is forwarded to the next VNF or to an OP (through an OLL connection). The OLL and ILL elements are specified either by an existing VNF ID or boundary node ID. In the case of an existing VNF, one of its interfaces is employed (which indicated in the corresponding MAC field). In the case of boundary nodes, no MAC is defined because the sender and receiver hosts are outside NIEP.

Figure 5 SFC simplified JSON schema

```
{
  "type": "object",
  "properties": {
    "ID": {"type": "string"},
    "VNFS": {"type": "array",
      "items": {
        "type": "object",
        "properties": {
          "ID": {"type": "string"},
          "PATH": {"type": "string"}
        }
      }
    },
    "IP": {"type": "object",
      "properties": {
        "ID": {"type": "string"},
        "LINK": {"type": "string"}
      }
    },
    "OPS": {"type": "array",
      "items": {
        "type": "object",
        "properties": {
          "ID": {"type": "string"},
          "LINK": {"type": "string"}
        }
      }
    },
    "CONNECTIONS": {"type": "array",
      "items": {
        "type": "object",
        "properties": {
          "OLL": {"type": "string"},
          "ILL": {"type": "string"},
          "OLL_MAC": {"type": "string"},
          "ILL_MAC": {"type": "string"}
        }
      }
    }
  }
}
```

The complete topology is represented with a third schema, shown in Figure 6. The five attributes of this description carry information about VNFs and SFCs, plus the Mininet emulated network infrastructure. The NIEP topology is identified with a unique ID. A NIEP instance is responsible for the execution of a topology, thus after the topology is deployed the ID also represents the NIEP process itself.

The virtual functions and function chains of a given NIEP topology are defined by the VNFS and SFCS properties. These attributes specify the location of the corresponding description files, created according to the schemas presented above. Note that the VNFs used in an SFC specified with the SFCS attribute should no be explicit in the VNFS attribute. Whenever a duplicated request is required, no ID should be replicated, (i.e., the internal SFC ID must be different from the VNF ID).

The Mininet network is described within a JSON object within the MININET attribute. This “sub-object” contains four attributes, each specifying an operational element of the emulation. The HOST attribute is an array that keeps virtual host IDs; the SWITCHES attribute is an array with the IDs of switches; The other two attributes, CONTROLLERS and OVSWITCHES, refer to the OpenFlow SDN network. The CONTROLLERS attribute contains a list of controllers; each object of this list consists of the ID, controller IP address and the PORT the controller uses to communicate. The configuration and initialisation of the OpenFlow controller are out of the scope of NIEP, which is controller-agnostic, even though POX (Kaur et al., 2014) is used as default. The last attribute OVSWITCHES is another object array, which specifies the IDs and connections of OpenFlow switches. Any ID employed in the system must be unique.

Figure 6 Topology simplified JSON schema

```
{
  "type": "object",
  "properties": {
    "ID": {"type": "string"},
    "VNFS": {"type": "array",
      "items": {"type": "string"}
    },
    "SFCS": {"type": "array",
      "items": {"type": "string"}
    },
    "MININET": {
      "type": "object",
      "properties": {
        "HOSTS": {"type": "array",
          "items": {
            "type": "object",
            "properties": {
              "ID": {"type": "string"},
              "IP": {"type": "string"}
            }
          }
        },
        "SWITCHES": {"type": "array",
          "items": {"type": "string"}
        },
        "CONTROLLERS": {"type": "array",
          "items": {
          "type": "object",
            "properties": {
              "ID": {"type": "string"},
              "IP": {"type": "string"},
              "PORT": {"type": "string"}
            }
          }
        },
        "OVSWITCHES": {"type": "array",
          "items": {
          "type": "object",
            "properties": {
              "ID": {"type": "string"},
              "CONTROLLER": {"type": "
                  string"}
            }
          }
        }
      }
    },
    "CONNECTIONS": {"type": "array",
      "items": {
        "type": "object",
        "properties": {
          "IN/OUT": {"type": "string"},
          "OUT/IN": {"type": "string"},
          "IN/OUTIFACE": {"type": "string"}
              ,
          "OUT/INIFACE": {"type": "string"}
        }
      }
    }
  }
}
```

Finally, the CONNECTIONS attribute is used to specify the interconnections of Mininet components among themselves and with VNFs and SFCs. The CONNECTION JSON object has two mandatory components: IN/OUT and OUT/IN. The IN/OUTIFACE and OUT/INIFACE indicate the virtual interface where the VNF will set up a connection. In the context of SFCs, the connection is specified in terms of its VNFs, typically one connection is defined for input and one for output. The connections between the VNFs of an SFC are specified in the SFC description file.

5 Case study and experimental evaluation

In this section, we describe a case study executed to obtain empirical results to evaluate the effectiveness of NIEP. First, in Sub-section 5.1, the case study described. Next, in Sub-section 5.2, results are presented and discussed. Finally, a qualitative evaluation is discussed in Sub-section 5.3. The main objective of these experiments is to investigate whether our platform can efficiently and effectively emulate heterogeneous NFV scenarios deployed on different network topologies. We also assess whether/how NIEP meets the requirements defined in Section 3.

5.1 Case study: description

The experimental setup defined is composed of two locations: the Customer Premises (CP) and an Internet Service Provider (ISP), as shown in Figure 7. In the CP, a Mininet host acting as a client is connected to a VNF with limited resources (1 core, 192 MB 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, 2 GB 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.

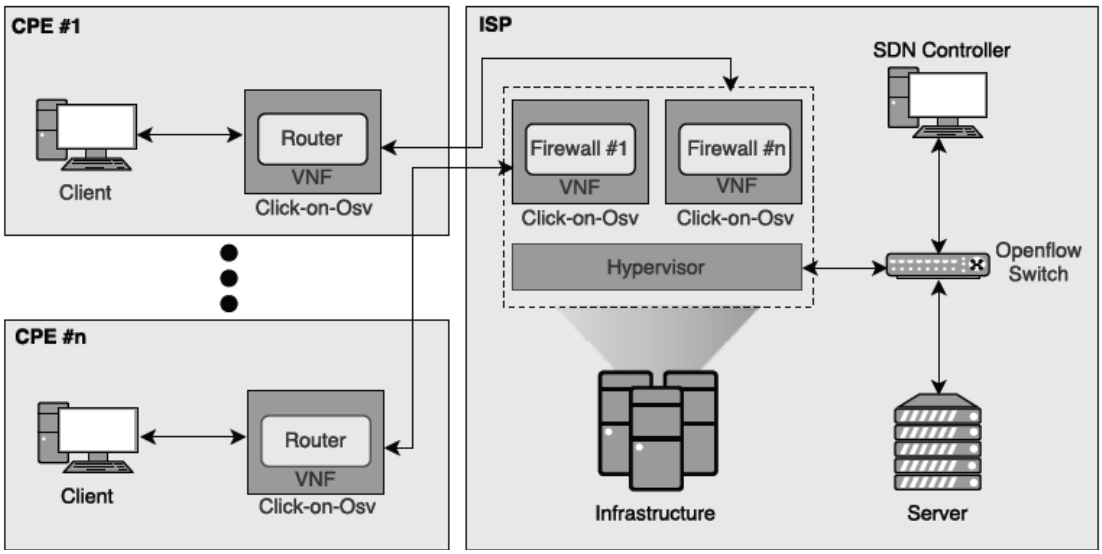
Four different configurations were used to evaluate how the number of customers connected to the ISP impacts the performance of NIEP. We varied the number of CPEs from 2, 4, 8 and 16. In addition, a second configuration was tested, in which two VNFs implementing a firewall were deployed on the ISP side with the purpose of balancing the load imposed by the CPEs.

The experiments were executed on the following system. The CP instances were executed on an Intel Core i7-6700k@4.00 GHz server, with 8GB RAM DDR4, 4 cores, and running CentOS 7. The ISP, in turn, was executed on an Intel Xeon E3-1220v6@3.00 GHz, 8GB RAM DDR4, 4 cores, running Ubuntu 14.04. The hosts were connected on a 1Gbps Ethernet network. We employed the KVM hypervisor to deploy both the Mininet VM and VNFs in both hosts.

5.2 Results

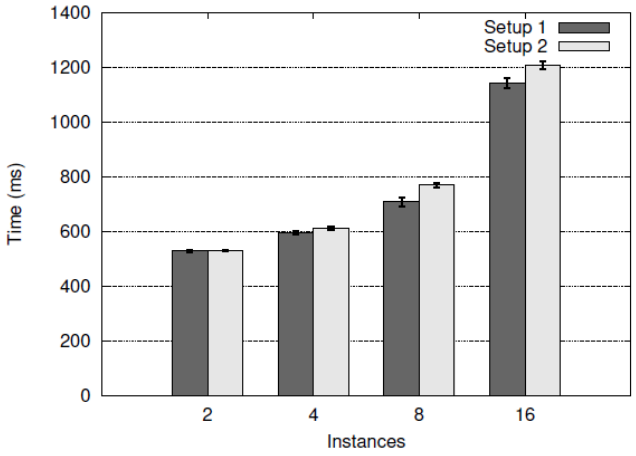
Each experiment was repeated 30 times, results are presented with a confidence interval of 99%. NIEP can be used to emulate multiple NFV scenarios, providing fine-grained control over several configuration parameters. The network topology can be easily changed, for example, to test different network paths, or to add, remove, and reconfigure hosts, or also to change link properties. The boot time is an important metric to be evaluated since it can impact the time it takes to evaluate a configuration.

Figure 7 Experimental evaluation: environment



The NIEP components were instrumented to send the boot time to a centralised server. The longest boot time corresponds to the VNFs on the CP side, as their number grows this increases the load on the processor. At the same time, the number of VNFs running on the processor used to emulate the ISP does not change as a single experiment is executed, it only changes from one experiment to another. The VNFs are instantiated at the same time, and Mininet is started before that. Thus, the boot time is the time Mininet takes to initialise plus the average boot time of the VNFs of the CP side. To make it clear, this is the time it takes for the entire emulation setup to be ready to execute. Results are shown in Figure 8.

Figure 8 NIEP evaluation: average boot time



In the second experiment, two VNFs implementing firewalls were deployed instead of a single one. Because of this, the boot time is slightly higher on the CPs VNFs, as they need to do some additional configuration to send traffic to the two different firewalls. The exception is when two CP instances are deployed (as shown in the first two bars of Figure 3) – the first setup takes 529 ms plus/minus 3 ms, and the second setup 2 takes 532 ms plus/minus of 2.8 ms. This happens because with only two CP instances there are still free processor cores left that are used exclusively by the hypervisor and operating system to execute tasks related to the configuration and deployment of virtual machines.

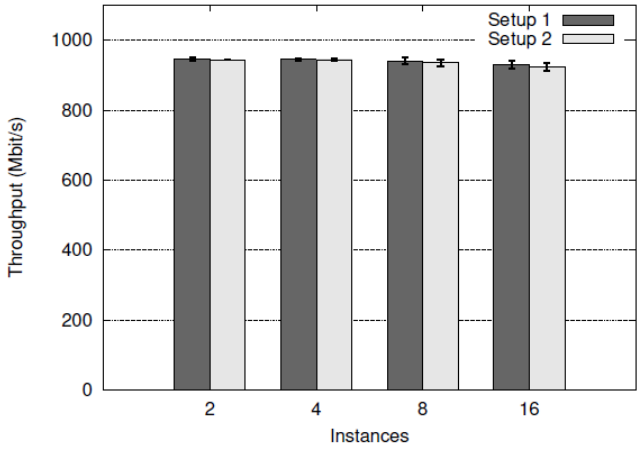
The evaluation of the performance of NIEP under heavy load allows the identification of bottlenecks caused by different factors on the emulation, due for instance to CPU and memory limitations. Thus if for example, physical resources are not sufficient, this would be reflected in the low throughput between the CPs and the ISP, due to the limited packet processing capacity. In this case, the throughput gets much lower than link speed (1 Gbps). *iperf* (Tirumala et al., 2005) was employed in the experiments; all CPs send traffic at the same time to the server running at the ISP side. The values obtained from each CP in each experiment were aggregated since CPs share the same 1 Gbps link with which they are connected to the ISP.

As shown in Figure 6, in all experiments executed for both setups the bottleneck was always the link between the CPs and the ISP. In this way, increasing the number of instances does not affect link throughput. We can conclude that NIEP scales well as required.

5.3 Discussion

Regarding the scalability, NIEP proved to be able to emulate complex scenarios, with increasing numbers of hosts, switches, and VNFs, as well as the overall topology. The good scalability can be explained because in NIEP the VNFs do not run within Mininet. Instead, the VNFs are connected through external bridges, which allow their execution remotely. On the other hand, although EsCAPE can also simulate complex scenarios, VNFs are deployed on the same host, since containers are defined within Mininet and connections must be established locally.

Figure 9 NIEP evaluation: throughput



Different from Mininet, the topologies NIEP employs are defined at a high level with JSON, which simplifies the process for deploying the infrastructure. Users can define, in addition to hosts, switches, and controllers, different types of VNFs, different amounts of resources that are allocated for the VNFs, and the connections among them, including the capacity of creating SFCs.

The use of hypervisor-based virtualisation 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 can be a serious limitation for the use of the other platforms discussed in Subsection 2.2, which rely on container-based virtualisation.

Finally, NIEP is more secure that the other platforms as NIEP is based on VMs while the others are based on containers which have more vulnerabilities.

6 Conclusions and future work

The Network Function Virtualisation (NFV) paradigm replaces traditional middleboxes with virtual functions that are executed on general purpose hardware. NFV brings multiple advantages

in terms of cost and flexibility, but it also brings new challenges. In this work, we presented NIEP, an NFV Infrastructure Emulation Platform to emulate VNFs. Emulation platforms provide a realistic alternative to execute VNFs. This NIEP allows VNFs to be tested and evaluated before they are deployed in production networks. Most existing NFV emulation platforms are based on containers or process virtualisation. Furthermore, they do not provide native support for the distribution of the emulation, which should be based on processes running and communicating on different machines. Thus the emulation is limited to a single machine, which has obvious scalability limitations.

NIEP is based on Click-on-OSv and Mininet. NIEP is based on VMs, which provides higher security guarantees that containers. NIEP allows the emulation of different NFV scenarios and VNF design and evaluation, supporting the emulation of heterogeneous infrastructures and scenarios. The evaluation of the performance of NIEP included the boot time and the throughput of VMs running CP and ISP sites. The results show that the boot time of VNFs increases almost linearly, indicating almost no impact of NIEP on VNF instantiation. Moreover, increasing the number of VNFs does not reduce throughput.

Future work includes the design of a user-friendly web interface for network operators, allowing information about VNFs and their operation to be obtained with a REST API. Furthermore, it is important to extend the tool to support different VNF technologies, such as nginx and BRO.

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

Baumgart, I., Heep, B. and Krause, S. (2007) 'Oversim: a flexible overlay network simulation framework', *Proceedings of the IEEE Global Internet Symposium*, IEEE, pp.79–84.

Bernardos, C.J., Contreras, L.M., Vaishnavi, I., Szabo, R., Li, X., Paolucci, F., Sgambelluri, A., Martini, B., Valcarenghi, L., Landi, G., Andrushko, D. and Mourad, A. (2018) 'Multi-domain network virtualization', *Internet-Draft draft-bernardos-nfvrg-multidomain-04*, pp.1–15.

Carson, M. and Santay, D. (2003) 'Nist net: a linux-based network emulation tool', *ACM SIGCOMM Computer Communication Review*, Vol. 33, No. 3, pp.111–126.

Chiosi, M. and Wright, S. et al. (2012) Network functions virtualisation (nfv)', *ETSI NFV ISG, White Paper*, Vol. 1.

Garay, J. Matias, J. Unzilla, J. and Jacob, E. (2016) 'Service description in the nfv revolution: trends, challenges and a way forward', *IEEE Communications Magazine*, Vol. 54, No. 3, pp.68–74.

GS NFV (2013) *GS NFV 001: Network Functions Virtualisation (nfv)*; Group Report, France.

Halpern, J. and Pignataro, C. (2015) 'Service function chaining (sfc) architecture', *Internet Engineering task Force*. Doi: 10.17487/RFC7665.

Han, B., Gopalakrishnan, V., Ji, L. and Lee, S. (2015) 'Network function virtualization: challenges and opportunities for innovations', *IEEE Communications Magazine*, Vol. 53, No. 2, pp.90–97.

Handley, M. (2006) 'Why the internet only just works', *BT Technology Journal*, Vol. 24, No. 3, pp.119–129.

Imran, M., Said, A.M. and Hasbullah, H. (2010) 'A survey of simulators, emulators and testbeds for wireless sensor networks', Proceedings of the *2010 International Symposium on Information Technology*, Vol. 2, pp.897–902.

Karl, H., Dräxler, S., Peuster, M., Galis, A., Bredel, M., Ramos, A., Martrat, J., Siddiqui, M.S., Rossem, S.V. and Tavernier, W. et al. (2016) 'Devops for network function virtualisation: an architectural approach', *Transactions on Emerging Telecommunications Technologies*, Vol. 27, No. 9, pp.1206–1215.

Kaur, S., Singh, J. and Ghumman, N.S. (2014) 'Network programmability using pox controller', Proceedings of the *ICCCS International Conference on Communication, Computing and Systems, IEEE*, Vol. 138, pp.1–5.

Kohler, E., Morris, R., Chen, B., Jannotti, J. and Kaashoek, M.F. (2000) 'The click modular router', *ACM Transactions on Computer Systems (TOCS)*, Vol. 18, No. 3, pp.263–297.

Lantz, B., Heller, B. and McKeown, N. (2010) 'A network in a laptop: rapid prototyping for software-defined networks', *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, pp.1–6.

Lu, K., Liu, S., Feisullin, F., Ersue, M. and Cheng, Y. (2015) 'Network function virtualization: opportunities and challenges', *IEEE Network*, Vol. 29, No. 3, pp.4–5.

Marcuzzo, L da C., Garcia, V.F., Cunha, V., Corujo, D., Barraca, J.P., Aguiar, R.L., Schaeffer-Filho, A.E., Granville, L.Z. and dos Santos, CRP. (2017) 'Click-on-osv: a platform for running click-based middleboxes', *Proceedings of the IFIP/IEEE Symposium onIntegrated Network and Service Management (IM)*, IEEE, pp.885–886.

Mijumbi, R., Serrat, J., Gorricho, J-L., Bouten, N., Turck, F.D. and Boutaba, R. (2016) 'Network function virtualization: state-of-the-art and research challenges', *IEEE Communications Surveys and Tutorials*, Vol. 18, No. 1, pp.236–262.

Mininet Team (2012) *Mininet: An instant virtual network on your laptop (or other pc)*, Information and Networking Technologies Research and Innovation Group.

Mohallel, A.A., Bass, J.M. and Dehghantaha, A. (2016) 'Experimenting with docker: Linux container and base os attack surfaces', *Proceedings of the International Conference on Information Society (i-Society)*, pp.17–21.

Morabito, R., Kjällman, J. and Komu, M. (2015) 'Hypervisors vs. lightweight virtualization: a performance comparison', *Proceedings of the IEEE International Conference on Cloud Engineering*, pp.386–393.

Morris, R., Kohler, E., Jannotti, J. and Kaashoek, M.F. (1999) 'The click modular router', *ACM SIGOPS Operating Systems Review*, Vol. 33, No. 5, pp.217–231.

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

Pfaff, B., Pettit, J., Koponen, T., Jackson, E.J., Zhou, A., Rajahalme, J., Gross, J., Wang, A., Stringer, J. and Shelar, P. et al. (2015) 'The design and implementation of open vswitch', *Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation*, pp.117–130.

Salopek, D., Vasić, V., Zec, M., Mikuc, M. Vašarević, M. and Končar, V. (2014) 'A network testbed for commercial telecommunications product testing', *Proceedings of the 22nd International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pp.372–377.

Schaeffer-Filho, A., Mauthe, A., Hutchison, D., Smith, P., Yu, Y. and Fry. M. (2013) 'Preset: a toolset for the evaluation of network resilience strategies', *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM'13)*, IEEE, pp.202–209.

Sonkoly, B., Czentye, J., Szabo, R., Jocha, D., Elek, J., Sahhaf, S., Tavernier, W. and Risso, F. (2015) 'Multi-domain service orchestration over networks and clouds: a unified approach', *Proceedings of the ACM Conference on Special Interest Group on Data Communication* (SIGCOMM'15), ACM, New York, NY, USA, pp.377–378.

Tirumala, A., Qin, F., Dugan, J., Ferguson, J. and Gibbs, K. (2005) *Iperf: The tcp/udp bandwidth measurement tool*. Available online at: http://iperf.fr

Varga, A. and Hornig, R. (2008) 'An overview of the omnet++ simulation environment', *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems and Workshops*, Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, Marseille, France.

Wette, P., Draxler, M., Schwabe, A., Wallaschek, F., Zahraee, M.H. and Karl, H. (2014) 'Maxinet: distributed emulation of software-defined networks', *Proceedings of the Networking Conference*, IEEE, Norway, pp.1–9.

Notes

1 Available online at: https://github.com/ViniGarcia/NIEP

2 Available online at: https://github.com/containernet/containernet

3 Available online at: http://www.docker.org/

4 Available online at: https://libvirt.org/

5 Available online at: https://www.linux-kvm.org/page/Virtio