EasyChair Preprint

№ 681

Simulating Software Defined Networking Using Mininet to Optimize Host Communication in a Realistic Programmable Network.

Lindinkosi Zulu, Kingsley A. Ogudo and Patrice Umenne

December 14, 2018

Simulating Software Defined Networking Using Mininet to Optimize Host Communication in a Realistic Programmable Network.

Abstract— **In this paper, Mininet-WiFi was used to simulate a Software Defined Network to demonstrate Mininet-WiFi' s ability to be used as the Software Defined Network emulator which can also be integrated to the existing network using a Network Virtualized Function. A typical organization's computer network was simulated which consisted of a website hosted on the LAMP (Linux, Apache, MySQL, PHP) virtual machine, and an F5 application delivery controller (ADC) which provided load balancing of requests sent to the web applications. A website page request was sent from the virtual stations inside Mininet-WiFi. The request was received by the application delivery controller, which then used round robin to send the request to one of the web servers on the LAMP virtual machine. The web server then returned the requested website to the requesting virtual stations using the simulated virtual network. The significance of these results is that it presents Mininet-WiFi as an emulator, which can be integrated into a real programmable networking environment offering a portable, cost effective and easily deployable testing network, which can be run on a single computer. These results are also beneficial to modern network deployments as the live network devices can also communicate with the testing environment for the data center, cloud and mobile provides.**

Keywords—SDN; Mininet; Mininet-WiFi; 802.11, Virtual machine, Artificial Intelligence, Internet protocol, NVF, VNF

I. INTRODUCTION

Moving to Software Defined-based networking is not without its challenges. Converting from a proprietary to an open system involves more moving parts, including controllers, clients, orchestration systems, and business applications. In many cases, Software Defined Networking (SDN) components must interact with legacy components introducing further complexity. This has created a room for testbeds which can be used to test the SDN network before being implemented in real network to minimize down time and to provide a test environment which is close to real world where issues can be identified and rectified. A suitable tool or emulator, which must produce close to real live situation, must be used to obtain outcomes, which can be implemented as is in real world.

Traditional Internet Protocol (IP) networks are complex and many enterprises find it a challenge to manage as network operators need to configure each individual network device separately and most often having to use vendor-specific commands. This is because each device has it own control, management and forward planes.

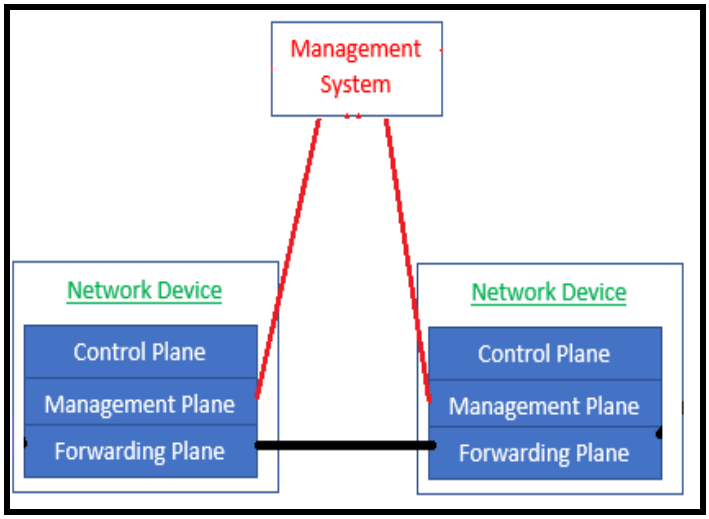

Fig. 1. Traditional network device

SDN gives hope to address these challenges which current network infrastructures are not able to address. It does so by separating the network's control logic from routers and switches that forward the traffic. It also separates the control and data planes leaving network switches being simple forwarding devices and the control logic is implemented in a logically centralized controller [1].

Software defined networking (SDN) is the physical separation of the network control plane from the forwarding plane, where a control plane can control several devices. This creates a three-layer architecture, which are infrastructure, control and application layers [2]. SDN makes it easier for network operators to evolve network capabilities. Before, network devices used to use closed proprietary features but with SDN, a single software program can control the behavior of the entire network [3].

This intelligence makes it possible to offer networking-as-a-Service (NaaS), which significantly reduces expenses both capital and operational and enables fast service architecture. This is due to the fact that the data plane is highly programmable from the remote-control plane at the controlling application [4]. SDN also offers enhanced configuration, improved performance and encouraged innovation [5].

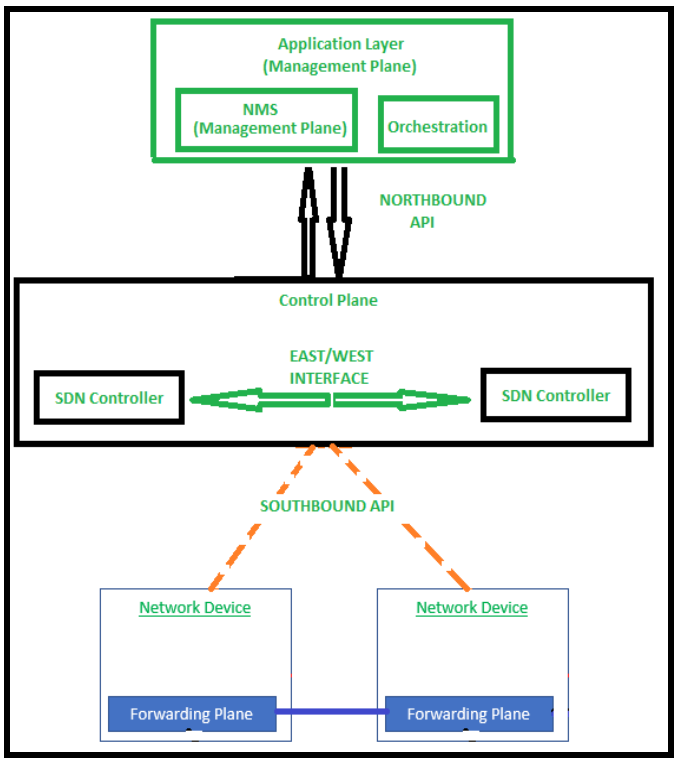

Fig. 2. Software Defined network device communication

The high demand for data has affected the telecommunication industry, more specially the mobile network providers. This hunger for data is one of the catalysts for connectivity speeds of 5G networks. Service providers are facing challenges in complying with connectivity demands without substantial financial investments [6]. To address this issue, the industry had to look for initiatives aiming at cost reduction, increase of network scalability and service flexibility. The two networking architectures introduced to meet these requirements are Network Functions Virtualization and Software Defined Networking [7].

Software Defined Networking is classically defined as the separation of the control plane from the forwarding plane where the control plane is centralized while Network Functions Virtualization is the virtualization of services instead of using the hardware purposefully built to provide that service.

Network Functions Virtualization is a framework defined by the European Telecommunications Standards Institute (ETSI) that specifies the virtualization of various network services such as firewalls, load balancers and any other services typically associated with dedicated purpose-built hardware. In the telecommunication industry, NFV proposes to run the mobile network functions as software instances on commodity servers or datacenters, while SDN supports a decomposition of the mobile network into control-plane and data-plane functions. The combination of both SDN and NFV is considered as a very promising combination in achieving a cost efficient mobile network architecture within the mobile network environment [8].

NFV has been proposed as a model that resolves the functions of placement and aims at minimizing the transport network load overhead against several parameters such as data-plane delay, number of potential data centers and SDN control overhead. By moving network appliance functionality from proprietary hardware to software, Network Function Virtualization promises to bring the advantages of cloud computing to network packet processing [9]. It is for this reason that this paper looks to Mininet and its wireless extension Mininet-WiFi as the emulator, which can be used to emulate a Software Defined Network, intergraded on the network using a Network Virtualized Function (NVF) in the form of an application delivery (load balancer).

A. Mininet

Recent Software-Defined Networking (SDN) approaches propose new means for network virtualization and programmability advancing the way networks can be designed and operated, including user-defined features and customized behavior at run-time [10]. The need for fault tolerance and scalability is leading to the development of distributed Software Defined Networking operating systems and applications. These developments in innovations calls for an emulator, which will be able to produce reliable results when emulating such networks [11].

Mininet provides the platform to understand how actual Software Defined Networking works by creating a virtual network similar to the real network. This can be applied to run on small as well as very large-scale networks. One of the advantage of using Mininet is that, an application that works on it can be easily deployed to a real network [12].

Mininet is a network rapidly prototyping system which support, testing and simulation of Software Defined networks. The greatest value of Mininet is supporting collaborative network research by enabling self-contained Software Defined Network prototypes, which anyone with a personal computer (PC) or laptop can download, and use [13]. To achieve this, Mininet uses lightweight approach of OS-level virtualization features ranging from processes and network namespaces, which make it possible to scale to hundreds of nodes and represents a qualitative change in workflow through its ability to run and debug in real-time [14].

Among the main reasons for emulating, a network is to be able to test and prove concepts. Current information on Software Defined Networking can be found in research papers and in white papers [15]. In the case that an organization, needs to prove these concepts or plans to deploy Software Defined Networks, the results can be easily reproduced. Mininet enables virtual network systems, where an environment of virtual hosts, switches, and links runs on a modern multicore server, using real application and kernel code with software-emulated network elements. An experiment has been conducted using Mininet to reproduce key results from published network experiments such as DCTCP, Hedera, and router buffer sizing which were successfully reproduced highlighting another important ability of Mininet [16].

Mininet is not the only simulator, which can be used to simulate SDN networks. Other specialized hardware network devices require specialized programming languages [17] to run. Other notable simulators available include the used of Raspberry-Pi [18] to develop a cost-effective OpenFlow testbed for a small scale SDN networking and Fs-SDN [19]. Although some of these other simulators do have advantages over Mininet on some aspects of simulation [20], Mininet remains the simulator of choice for SDN networks due to its flexibility and many advantages. Other simulators use full system virtualization, heavyweight containers with increasing complexity and overheads while decreasing usability. Mininet support the development of SDN systems and applications reliably without access to an expensive testbed [21].

Networks emulated in Mininet have produced reliable results, which has made Mininet to be used as a reference system when other emulators like Fs-SDN were being developed or tested. Fs-SDN is a Python-based tool developed for generating network flows records and interface counters. To evaluate fs-SDN accuracy, scalability, and speed, a side by side setting up between fs-SDN and Mininet was done with a series of identical network and traffic configurations. After which network traffic and system-level measurements where compared between the two. In this investigation, Mininet was discovered to be a better tool.

B. Mininet-WiFi

Mininet can also be used to emulate Software Defined Wireless Networks. To achieve this, the base code of Mininet must be extended by modifying classes and scripts to support wireless functionalities while also keeping all Software Defined Networking capabilities from the standard Mininet network emulator. Mininet-WiFi is a fork of Mininet emulator, which extends its functionality by adding virtualized Wi-Fi stations and access points based on the standard Linux wireless drivers and the 802.11_hwsim wireless simulation driver. It adds classes to support the addition of wireless devices in a Mininet network scenario and to emulate the attributes of a mobile station such as position and movement relative to the access points. The 802.11 is the wireless standard by Institute of Electrical and Electronics (IEEE), which provides specifications for implementing wireless communications using the Wi-Fi (Wireless Fidelity). [22]

Mininet-WiFi developers have showcased it in a scenario with ad hoc and infrastructure wireless modes using a single experimental platform integrating virtual and physical nodes. This demonstration featured Mininet-WiFi as an emulator with the ability to run realistic experiments in hybrid physical-virtual environments, where users were able to experience it first hand by connecting their devices and interacting with virtual Wi-Fi stations in a wireless mesh network. They were able to connect to the internet through the emulated Software Defined Wireless Network infrastructure. Mininet-WiFi enhances Mininet emulator with virtual wireless stations and access points while keeping the original SDN capabilities and the lightweight virtualization software architecture [23].

II. METHODOLOGY (NETWORK DESIGN)

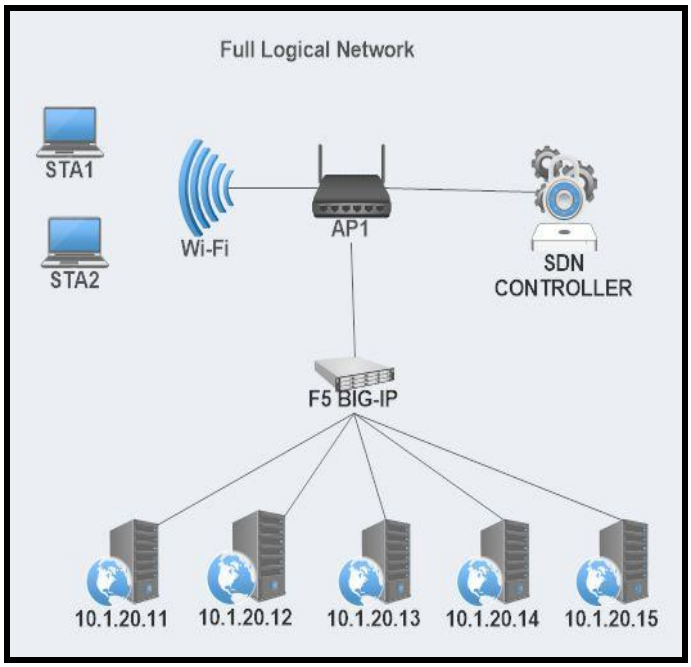

Fig. 3. Logical network simulated on this paper

The network consists of website hosted on the LAMP (Linux, Apache, MySQL, PHP) virtual machine, an F5 application delivery controller (ADC) to provide load balancing of requests sent to the web applications and Mininet-WiFi to simulate a Software Defined wireless network. All these network components are integrated to form a typical computer network of an organization.

A. LAMP Server

LAMP is a group of open source software used to setup and run webservers. The acronym stands for Linux, Apache, MySQL, and PHP. This is because it uses Linux as the

operating system, Apache as the Web server, MySQL as the relational database management system and PHP as the object-oriented scripting language.

LAMP is mostly referred to as a LAMP stack because it has four layers. This stack can be built on different operating systems and the acronym changes to reflect that operating system. For an example, with Windows operating system, it is called WAMP; with Macintosh system, it is called MAMP; and with a Solaris system, it is called SAMP.

For this simulation, LAMP server virtual machine with five (5) websites already configured was downloaded from F5 and integrated to this SDN network. The configured servers' names are from server 1 to server 5 with Internet Protocol (IP) addresses 10.1.20.11 up to 10.1.20.15. All these configured servers deliver the same web application.

This already configured LAMP was chosen because of its ability to simulate real world web application as deployed by many organizations.

B. F5 BIG-IP Virtual Edition (VE)

F5 is a company that specializes in application delivery networking (ADN) technology. It is involved in the delivery of web applications, security, performance, availability of servers, data storage devices, and other network and cloud resources.

To do this, F5 uses BIG-IP platform which is a blend of software and hardware which is a load balancer and a full proxy. It gives the ability to control the traffic that passes through the network and comes in two (2) forms, physical hardware and virtual Edition (VE).

The virtual editions of BIG-IP products offer the same variety of features available in hardware solutions and can be deployed on a public or private cloud.

The company name F5 was inspired by the 1996 movie Twister in which reference was made to the fastest and most powerful tornado on the Fujita Scale: F5.
F5 founders believed that this company will cause the most powerful change in the networking field.

BIG-IP is not an acronym for anything either, F5 got the concept from TCP/IP (Transmission Control Protocol/ Internet Protocol) notation. The BIG part of the name is from F5 view of this technology as being a full proxy, which presents a virtual IP on behalf of many devices that are behind it. This makes it according to F5 an IP bigger that a normal IP address, hence it is called BIG-IP.

This simulation uses BIG-IP Virtual Edition (VE) which after downloading from F5 was licensed and then configured. The configured network features include the internal VLAN, which is the network part that communicates with backend servers running on the LAMP stack and the external VLAN, which is the network part that communicates with Mininet-WiFi.

On The BIG-IP application controller, a virtual server was created which is accessible only from the external VLAN and is used to request web applications from the backend servers. These are grouped on a pool program to provide load balancing to the LAMP virtual machine using round-robin load balancing algorithm.

C. Mininet-WiFi

A Mininet-WiFi virtual machine was created by installing Mininet-WiFi on an Ubuntu Server virtual machine. Two (2) interfaces were configured and associated with Mininet-WiFi virtual machine (VM).

The first interface was for management so that Mininet-WiFi can be accessed using any terminal emulator, which offers better command line interface, compared to Mininet-WiFi command prompt.

The second interface was then used to intergrade Mininet-WiFi onto the simulated network and was configured with the IP address on the same subnet as the external VLAN of the BIG-IP application delivery controller.

Mininet-WiFi basic network topology was used which consist of a wireless access point (AP1) with two wireless stations (Sta1, Sta2) and the controller. The access point is connected to a controller (C0) using virtual connection and the two (2) stations are attached to the access point (AP) using the simulated wireless interface.

Once the basic network was created, the external VLAN interface was then added to the access point (AP) interface using OpenFlow commands. Both stations (Sta1 and Sta2) internet protocol (IP) addresses were modified such that they are also on the same external VLAN subnet.

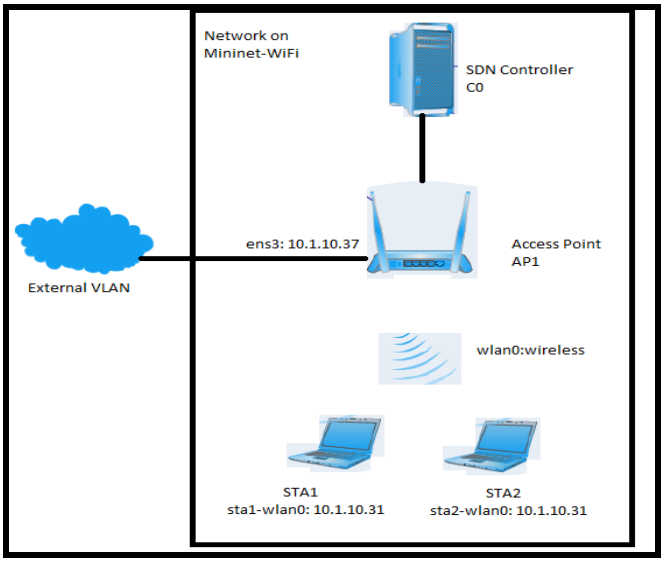

Fig. 4. Network inside Mininet-WiFi

III. RESULTS

To test our simulated network functionality, a hypertext terminal protocol (http) website page request was sent to the virtual server on the BIG-IP application delivery controller from the virtual stations (Sta1 and Sta2) connected to the access point (AP1) via simulated wireless interface inside Mininet-WiFi. The controller (C0) was able to create flow entries on the access point directing traffic to the external VLAN interface of the access point.

The request was received by the virtual server, which then used round robin to load balance the request to one of the web servers on the LAMP virtual machine. The web server returned the requested website in the hypertext terminal protocol (http) format to the requesting virtual stations using the simulated virtual network.

Fig 5 shows that from 0.016 s to about 0.032 s, a command to "GET" an (http) packet was issued from the virtual stations to the virtual server and then the (http) packet was received at the virtual stations signified by the "OK" message. If this is related to the information in fig 6 we notice that the round-trip time initially decreases from (0.016 – 0.027 s) indicating that no data is travelling during this time and then increases from (0.027 – 0.033 s), indicating that the (http) packet is being transmitted relative to time. Fig 7 shows that the average data throughput remains constant at 600 bits/s over a time period of (0.016 – 0.032 s) which means that there are no dropped packets during this period and that the data travels over the round trip without dropping any packets.

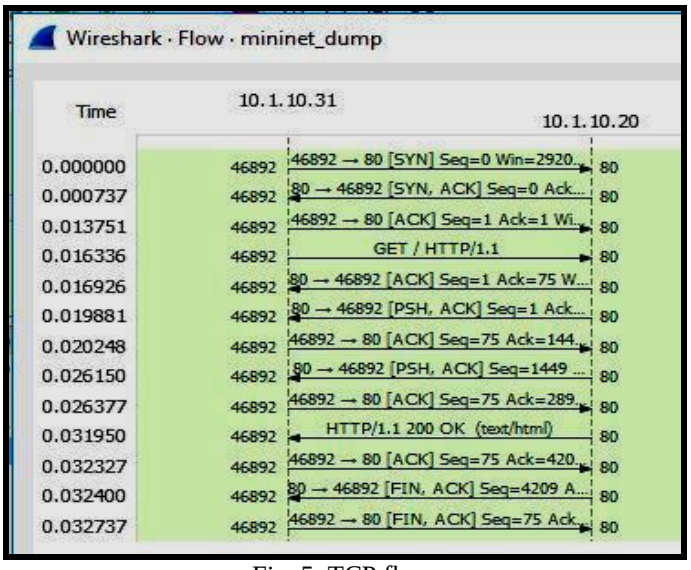

Fig. 5. TCP flow

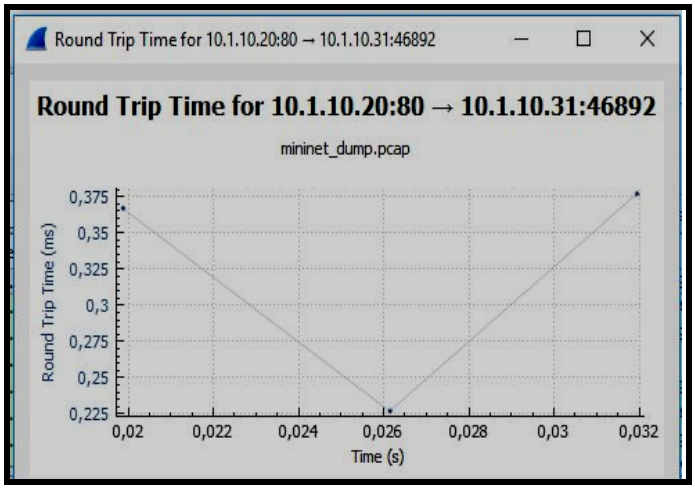

Fig. 6. Round Trip Time

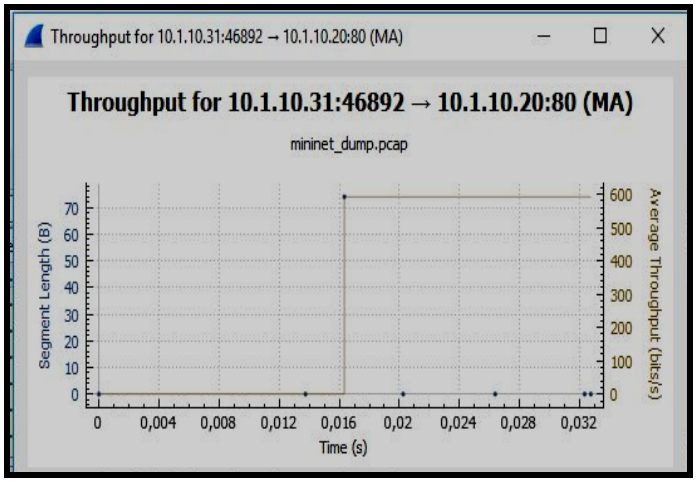

Fig. 7. Throughput

The result of this project is a successful communication between the Software Defined network devices inside the Mininet-WiFi emulator with the virtualized application delivery controller (load balancer) which is a Network Virtualized Function. This result shows Mininet and its wireless extension Mininet-WiFi as the suitable emulator which can be integrated into a realistic programmable networking environment using the combination of both SDN and NFV. These offer a portable, cost effective and easily deployable testing network, which can be run on a single computer.

CONCLUSION

Modern network technologies are moving away from traditional architecture, which is proprietary, and opting for more flexible and automated architectures, which support intelligent business models by allowing network programmability. The advent of Software Defined Networking and Network Functions Virtualization resulted in a need for an

emulator, which can be able to emulate networks even in realistic programmable networking environments.

Although developments of Mininet and it wireless extension Mininet-WiFi is an ongoing process, but results from this project are encouraging as it demonstrates that this emulator can be of benefit to even wireless network providers and organizations with virtualized network functions and cloud solutions. This project open doors for Mininet integration research with other technologies like Orchestrators, datacenters, Cloud, Business Support Systems (BSS) and Operations Support Systems (OSS).

REFERENCES

[1] Kreutz D, Fernando M, Ramos V, Esteves Verı´ssimo P, Rothenberg C. Azodolmolky S, Uhlig S. Software-Defined Networking: A Comprehensive Survey. Proceedings of the IEEE Vol. 103, No. 1, January 2015

[2] Open Networking Foundation (ONF). 2016. The SDN Solutions Showcase: Technical Report & Analysis version Number 1.0. Paper presented at SDN & OpenFlow world congress, Dusseldorf, Germany.

[3] Feamster N, Woodrow S, Sundaresan S, Kim H, Clark R. Voellmy A, 2012. The Past, Present, and Future of Software Defined Networking

[4] Lin Y, Pitt D, Hausheer D, Johnson E, Lin Y. 2014. Software-Defined Networking: Standardization for Cloud Computing Second Wave

[5] Xia W, Wen Y, Foh C, Niyato D, Xie H, 2015. A Survey on Software-Defined Networking.

[6] Hawilo H, Shami A, Mirahmadi M. 2014. NFV: state of the art, challenges, and implementation in next generation mobile networks (vEPC)

[7] Arsany Basta A, Kellerer W, Hoffmann M, Morper H, Hoffmann K. 2014. Applying NFV and SDN to LTE mobile core gateways, the functions placement problem

[8] Palkar S, Lan C, Han S, Jang K, Panda A, Ratnasamy S, Rizzo L, Shenker S.2015. A Framework for NFV Applications

[9] Matias J, Garay J, Toledo N. 2015. Toward an SDN-enabled NFV architecture

[10] Vitaly Antonenko V. 2013. Global Network Modelling Based on Mininet Approach.

[11] Handigol M, Heller B, Jeyakumar V, Lantz B, McKeown N. 2012. Reproducible Network Experiments Using Container-Based Emulation

[12] Lantz B, O'Connor B. 2015. A Mininet-based Virtual Testbed for Distributed SDN Development

[13] Lantz B, Heller B, McKeown N. 2010. A Network in a Laptop: Rapid Prototyping for Software-Defined Networks.

[14] Syrivelis D, Parisis G, Trosse D, Flegkas P, Sourlas V, Korakis T, Tassiulas L. Pursuing a Software Defined Information-Centric Network. Paper presented at the European workshop on Software Defined Networks in Darmstadt, Germany from 25-26 October 2012

[15] Frömmgeny A, Stohr D, Fornoffy J, Effelsberg W, Buchmanny A. 2016. Capture and Replay: Reproducible Network Experiments in Mininet

[16] Kumar D, Sood M. 2016. Software Defined Networks (S.D.N): Experimentation with Mininet Topologies

[17] Wang S. 2014. Comparison of SDN OpenFlow network simulator and emulators: EstiNet vs Mininet

[18] Weerawardhana J.L.M.N, Chandimal N.J.A.W, Bandaranayake A. 2015. SDN Testbed for Undergraduate Education

[19] Kim H, Kim J, Ko Y. 2014. Developing a Cost-Effective OpenFlow Testbed for Small-Scale Software Defined Networking

[20] Gupta M, Sommers J, Barford P. 2013. Fast, Accurate Simulation for SDN Prototyping

[21] Lantz B, O'Connor B. 2015. A Mininet-based Virtual Testbed for Distributed SDN Development.

[22] Fontes R, Afzal S, Brito S, Santos M, Rothenberg, C, 2015. Mininet-WiFi: Emulating software-defined wireless network.

[23] Fontes R, Rothenberg C. 2016. Mininet-WiFi: A Platform for Hybrid Physical-Virtual Software-Defined Wireless Networking Research