SENATUS: an experimental SDN/NFV Orchestrator

Sebastian Troia[(1)], Alberto Rodriguez[(1)], Rodolfo Alvizu[(1)], Guido Maier[(1)]

[(1)] *Politecnico di Milano, Dipartimento di Elettronica Informazione e Bioingegneria (DEIB), Italy*

***Abstract*—The fifth generation telecommunication standard (5G) will make use of novel technologies, such as Software Defined Networks (SDN) and Network Function Virtualization (NFV). New models are integrating SDN and NFV in a control plane entity responsible of the Management and Orchestration (MANO) of the whole system. This entity, acting as director of the 5G control plane, is known as Service Orchestrator. This work presents SENATUS, an experimental service orchestrator targeting research and testing environments. SENATUS implements a set of innovations to support the validation of the future network planner modules that will be integrated on the management entities of the 5G architecture. Furthermore, we present two testbed scenarios to show the capability of SENATUS to control the SDN and NFV technologies previously mentioned.**

Keywords—Software Defined Networking, Network Function Virtualization, OpenFlow, Openstack, Machine Learning

I. INTRODUCTION

Telecommunication networks have become a critical infrastructure for any society enabling economic growth and social prosperity [1]. The demand for data is surging rapidly every year. In 2021, the global IP traffic will reach 3.3 ZB (10^{21} bytes!) [2]. Cloud services and Machine-to-Machine (M2M) applications, capacity hungry applications, and bandwidth-hungry devices are contributing to the increment in traffic and data congestion. It is foreseen that the introduction of new technologies, like 5G, will boost this growth. This situation is forcing the underlying network technologies to change, increasing the level of programmability, control, and flexibility of configuration, while reducing the overall costs related to network operations [3]. Contrary to the evolution of previous generations, 5G will require not only improved networking solutions but also more sophisticated mechanisms for traffic differentiation to fulfil stronger end-to-end Quality of Service (QoS) requirements of numerous different "verticals", such as automotive, manufacturing, energy, eHealth, agriculture, etc [4] [5].

This work presents an experimental service orchestrator (SO) named SENATUS, targeting research environment, as devised to support testbeds for the development and validation of network services and network planning algorithms. It implements functions for managing SDN networks as well as for deploying network services on NFV infrastructure. In addition, SENATUS introduces an interface for horizontal communication that allows interworking with experimental Machine Learning (ML) modules and optimization algorithms. SENATUS is destined to provide a suitable environment for validating service orchestration functions and modules for network planning tools. The contribution of this paper can be summarized in four points:

1. Offer service orchestration in an infrastructure integrating SDN and NFV technologies.
2. Provide means for integrating experimental modules able to setup network configurations taking into consideration past and current situation of the network.
3. Propose testing SDN/NFV environments for SENATUS.
4. Supply a suitable set of metrics and data to perform performance evaluation.

SENATUS is presented together with two testbed architectures. The first one has its physical infrastructure emulated using Mininet and can validate the SDN-based functionalities, while the second, the main one, has been deployed using real equipment of BONSAI laboratory at Politecnico di Milano. The paper is organized as follows: Section II presents related works. Section III shows the SENATUS architecture. Section IV introduces two testbed architectures. Section V presents the validation and performance results of SENATUS. Finally, a discussion on open issues and the conclusions are presented in Section VI.

II. RELATED WORKS

This section will compare SENATUS orchestrator with other SO in the literature, highlighting similarities and innovations.

CORD [6], as one of the first implementations of service orchestration, introduced some of the basic concepts of the joint use of SDN and NFV. CORD uses the NFV for fast and flexible deployment of network services, while SDN interconnects the equipment and provides end-to-end QoS. CORD set the foundations for latter architectures, dividing the service orchestration into what we can consider: a network manager, an infrastructure manager and a higher entity (XOS) that takes the role of service Orchestrator. Similarly, to CORD, SENATUS uses OpenStack to provide a virtualization layer over the HW, and ONOS gives network data plane (DP) abstraction.

ADRENALINE [7][8] introduces the hierarchical approach to multi-domain orchestration that CORD lacked. It is a testbed designed to be integrated in advanced scenarios with multiple heterogeneous networks segments, each one with its own local manager that uses the Control Orchestration Protocol (COP) on its NBI to communicate with the multi-domain orchestrators. ADRENALINE is a powerful environment suitable for testing end-to-end 5G service orchestration. Statistics and status are pulled from the NBI of the bottom entities to the SBI of the superior one, and the actions and new configurations for the network are pushed from the top. Up to this point, ADRENALINE does not provide a horizontal communication interface like the one required by the network planner, that would allow the introduction of specific purpose modules to enhance the capabilities of the orchestration system. The lack on this interface makes it difficult to attach new modules to the service orchestrators without creating new layers of abstraction. By contrast, SENATUS introduces some horizontal functions that allows the delegation of some of

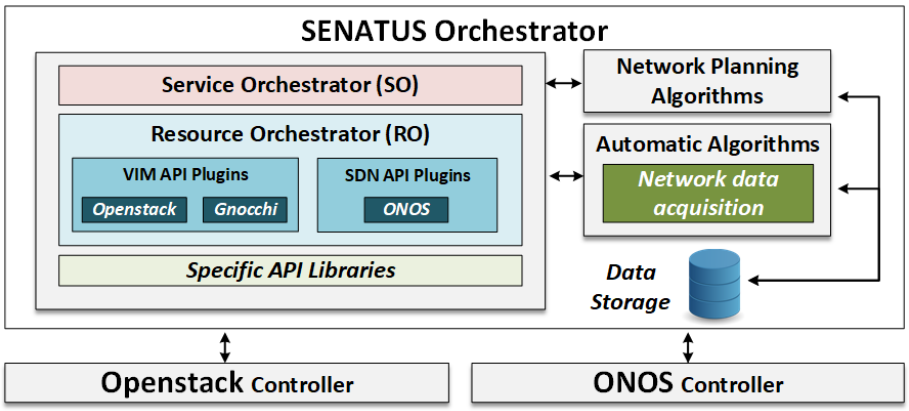

Fig. 1. CONTROL PLANE (CP) ARCHITECTURE WITH THE SENATUS ORCHESTRATOR

Table 1. SUMMARY OF ORCHESTRATOR COMMANDS

Resource Orchestrator (RO)	Service Orchestrator (SO)
List of VNF images	Upload VNF image
Get VNF image	Delete VNF image
List of servers	Create flavor
List of hypervisors	Create network
Devices on topology	Create snapshot
Host discovered	Get host traffic
List flows	Configure network

the decision-taking features to external Planner modules, such as ML routing solution or ML algorithms for VNF placement.

OSM [9] emphasizes multi-domain NFV/SDN orchestration, being devised to provide a single service orchestration point for both technologies from where to encompass the placement and interconnections, trough virtual overlay networks, of the VNFs forming the NS to be deployed. Nevertheless, OSM is still an immature project, and many of its planned features are not implemented yet. As an example, up to the date, it does not provide specific control over VNF placement and most of the SDN-related functions are yet to be integrated. Whereas the SENATUS orchestrator circumscribes to a less general scenario than OSM, that is, the control of a single administrative segment; the SO and the testbeds implemented on this work covert the full stack from the physical layer to the SO layer and, in addition, already implement the SDN features and VNF placement that OSM is still lacking.

ETSO [10] uses specific-purpose modules for VNF placement and interconnection. It implements the full stack, and can use several strategies for VNF placement due to their custom modules. However, despite following the TOSCA specifications for NS description, it is a self-contained orchestrator that does not follow the NFV standard and lacks compatibility to be integrated with common open solutions and projects.

The limitations reported in this section have been addressed developing SENATUS.

III. SENATUS ARCHITECTURE

SENATUS is a service orchestrator designed for research and validation environments. Its architecture resembles the structure proposed by the ETSI MANO framework and takes as reference the modular division used by OSM. SENATUS orchestrator corresponds to the highest layer of abstraction in the proposed Control Plane (CP) architecture, see Fig. 1. The name of SENATUS is borrowed from the ancient Roman Republic Senate, where administrative decision, usually taking into account the history, were made and passed down to the rest of the administrative system. On this line, we have reproduced this mechanism, combining the service orchestrator with horizontal modules that have the task of implementing actions on the underlying network. The CP architecture in Fig. 1 refers to the full system, which integrates five different entities:

1. Openstack [1] as Virtual Infrastructure Manager (VIM).
2. ONOS [11] as SDN controller.
3. SENATUS as the proposed SO.
4. Network Planning Algorithms collects the modules that provision the network services in the SDN network.
5. Automatic Algorithms serve as collectors of historical information needed to run the network planning algorithms.

Then, the CP is integrated with the Data Plane (DP) described in Section IV. Internal SENATUS orchestrator architecture divides its functionalities into two modules: Service Orchestrator (SO) and the Resource Orchestrator (RO). SENATUS orchestrator has been entirely programmed in Python.

A. Resource Orchestrator (RO)

SENATUS uses three sources as metric providers, which are: Openstack, Gnocchi and ONOS. Openstack has access to the information status of the system, such as characteristics of the deployed VNFs. Gnocchi [12] is a software that provides metric collection and aggregation at fixed time intervals and allows to define custom metrics regarding computing network resources. Gnocchi is especially useful for managing the storage of data coming from diverse sources, such as the case of distributed platforms like Openstack. ONOS, as SDN controller, is able to recover a wide set of metrics from the underlying network devices and provide them to its native applications and to external entities using the NBI. The granularity of the measures (by default 5 seconds) can be set in ONOS configuration. In contrast to Openstack and Gnocchi, where data is queried using dedicated API libraries, ONOS provides the data through REST API. Information from the REST APIs are transformed by SENATUS RO to JSON lists or matrices of data for being presented to the rest of the modules. Some of the information retrieved by the RO is included on Table 1.

B. Service Orchestrator (SO)

The SO module integrates the high-level control logic responsible of the Management and Orchestration (MANO). The set of functions provided by SENATUS SO (some of them presented in Table 1) often rely on a combination of data and actions to be taken by Openstack and ONOS. As

[1] Weblink: https://www.openstack.org/

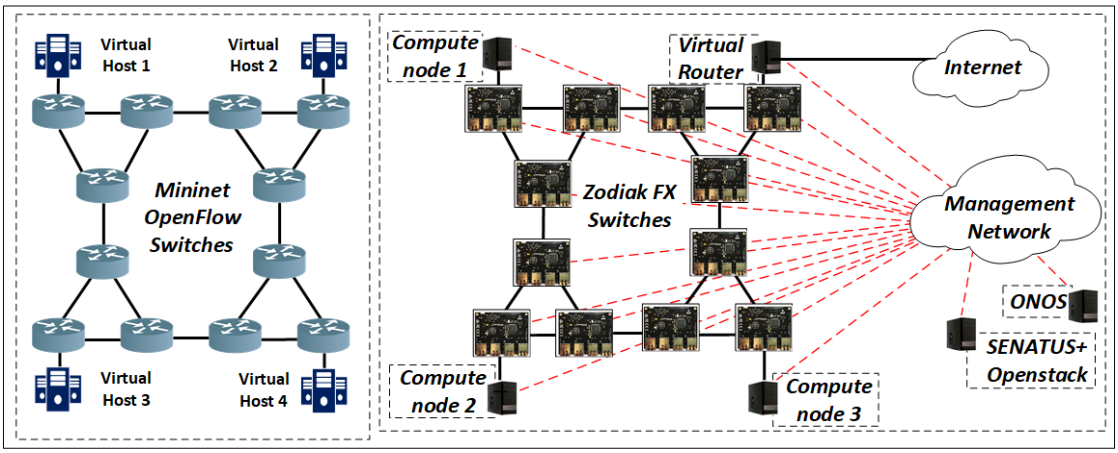

Fig. 2. TESTBED SCENARIOS. MININET TESTBED (LEFT) AND BONSAI TESTBED (RIGHT)

SENATUS RO abstracts the subjacent layers, the SO module focus on high level and complex processing functions not available at lower levels.

C. Network Planning Algorithms

This entity represents one of the main innovations proposed by this work. The Network Planning Algorithms encloses the algorithms that are executed to realize the network services, see Fig.1. This module gives the possibility in the future to implement different network optimization algorithms, such as VNF placement and routing computation. The algorithms defined in this module use the functions provided by the SO and RO, defined in the Table 1, to carry out their purpose. Network Planning Algorithms can access a data storage where it is possible to acquire historical data and therefore be able to train various machine learning based algorithms. In [13], a machine learning based algorithm has been implemented exploiting the potential of SENATUS. The algorithm classifies the traffic matrices in order to optimize routing in the underlying network scenarios (see Mininet and BONSAI testbed in section IV).

D. Automatic Algorithms

This entity is used to execute algorithms that collect data from the network and from servers running VNFs. The network data acquisition module is responsible for the continuous collection (every 5 seconds) and processing of information regarding network traffic exchanged between nodes and link status over time. This gives the possibility to the planning algorithms, given the history of the network, to optimize the distribution of network resources. Furthermore, it opens the possibility for a network operator to monitor his network and provide increasingly efficient services.

IV. TESTBED SCENARIOS

This section introduces the testbed scenarios that have been deployed to be used as experimental Data Plane (DP) working under the CP described. Two dedicated scenarios have been setup for SENATUS. The first one is devised for testing the SDN features of the service orchestrator. It does not have Openstack enabled and only takes advantage of the SENATUS functions related exclusively to ONOS. The topology of this first scenario is a virtual and emulated on Mininet: for this reason, this testbed will be named as "Mininet testbed" (see Fig. 2). The second environment is a physical implementation deployed at the Broadband Optical Networks, Security and Advanced Internet (BONSAI) laboratory of Politecnico di Milano. This second testbed, named "BONSAI testbed" (see Fig.2), provides both SDN and NVF capabilities, taking full use of the SENATUS. We are dedicating a last subsection to describe the VNFs used during this work to validate SENATUS.

A. Mininet Testbed

This testbed scenario has been created with a virtual topology emulated by Mininet [14] and an instance of ONOS version 1.12.0 Magpie. Mininet is one of the most widely used network emulators [15]. It can create virtual networks with operational nodes and allows interaction among them. One of the key features of Mininet is that it supports OpenFlow and Software-Defined Network (SDN) systems, which can be created as easy by running a single command. Mininet scenario is suitable for simplified testing of the SDN-related functions developed inside SENATUS. Using Mininet, the topology replicates 12 switches and 4 virtual hosts. These virtual hosts use the Distributed Internet Traffic Generator (D-ITG) tool to generate traffic [16]. D-ITG is a platform capable of producing packet-based traffic, emulating various stochastic processes for both IDT (Inter Departure Time) and PS (Packet Size) random variables (e.g., with Exponential, Uniform, Cauchy, Normal, Pareto, etc.). D-ITG supports both IPv4 and IPv6 traffic generation and it is capable to generate traffic at network, transport, and application layer.

B. BONSAI testbed

BONSAI testbed refers to the physical testing scenario in BONSAI laboratory at the Politecnico di Milano. This environment integrates equipment acting as VNF hypervisors along with a network composed OpenFlow switches[2]. OpenStack has been deployed for providing NVF capabilities, while ONOS is used as SDN controller. SENATUS is deployed as service orchestrator on the same computer where Openstack controller is hosted. Two separate networks are used in BONSAI testbed. Network 1 (red dotted lines in Fig.2) is a traditional IP network used for the communication exchange between the CP entities. Network 2 (black lines) is the SDN network composed by Zodiak FX Openflow switches. A set of computers, collectively named Compute Nodes, provide the equipment

[2] Weblink: https://northboundnetworks.com/products/zodiac-fx

and the power to run three instances of Openstack compute nodes, an instance of the Openstack controller with SENATUS and one with ONOS. A virtual router has been instantiated through OpenWrt VNF (see section IV.C.3) to connect both SDN and Management network, providing external connectivity to internet.

C. VNF deployed

Three VNFs have been used on the validation process of SENATUS.

1) CirrOS. It is a light Linux image typically used for proof of concepts and testing purposes. This Linux distribution only requires 13 MB of disk space, 32 MB of RAM and 1 CPU available to be launched, which makes it perfect for testing the health of the VIM installation.

2) Ubuntu 16.04. Ubuntu is a free open source Linux distribution operating system based on Debian. The VNF image is based on Ubuntu Server, requires around 279 MB of RAM, 1 CPU.

3) OpenWrt. It is a Linux distribution for embedded systems, typically used as an open source firmware for routers. As a proof of concept, we have generated and configured a OpenWrt VNF with 4 network interfaces and the necessary software packages to act as a virtual router. It requires 250 MB of RAM and 1 CPUs from the hypervisor.

V. Validation and Performance Results

In order to validate SENATUS, we have run several experiments aimed at evaluating:

1. *VNF Building time*: we measure the time required to deploy a VNF.
2. *VNF migration time*: we measure the time for creating a snapshot image. Then, we calculate the migration time as (VNF Building time + Snapshot creation time). We compare the data with results taken from literature.
3. *SENATUS overhead*: with this test we estimate the time taken by SENATUS to perform: a VNF deployment, a request of VNF status using Openstack API and a request of VNF metrics using Gnocchi API. We repeat the same experiments on the Command Line Interface and compare the performance.
4. *ONOS experiments*: this test is devised to measure the performance of SENATUS SDN capabilities on Mininet testbed.

A. VNF building time

With the first experiment, we measure the building time of a VNF inside the BONSAI testbed. For this, two of the VNFs already introduced in IV.C have been used: CirrOS and Ubuntu 16.04. We use several container flavors, as indicated by Table 2, to compare the outcoming values. The flavors are obtained by varying the RAM and the number of core assigned to the virtual machine container. We observe that assigning more RAM to the VNF containers the deployment time improves slowly. This is reasonable, since the processing of the installation of the VNF image inside the VM depends on the resources allocated for the VNF. However, taking into account the minimum size of RAM required by CirrOS (32 MB) and Ubuntu 16.04 (279 MB), the impact is minimal, only gaining a second for CirrOS deployment at the cost of allocating 20 times more RAM.

Table 2. CONTAINERS FLAVORS WITH VNF INSTATIATION TIME

Flavor	VCPU	RAM (MB)	VNF instantiation time (s)	
			Cirros	Ubuntu 16.04
F1	1	256	8.04	-
F2	1	512	7.87	7.72
F3	1	1024	7.57	7.66
F4	1	2048	7.54	7.47
F5	1	4062	7.02	6.93
F6	2	512	7.77	7.71

Deployment of Ubuntu VNF is speed up half a second using a container four times larger than the minimum size required. If we compare deployment times for flavors F2 and F6, we notice that allocating an extra CPU does not have significant impact.

B. Snapshot creation time and migration

Using the results of the VNF building time tests, we aim at checking the system performance for a theoretical VNF migration. We perform an offline migration, which involves two steps: generating the VNF snapshot and deploying the new VNF on the desired node. In Fig.3, we compared the migration time using some of the flavors defined in the previous section with results from literature. These results include experiments carried out migrating a VNF in a container equivalent to F4 in two high speed optical networks at 7 Gbits/s (SAI1) and 3.3 Gbits/s (SAI2) respectively [17]. We point out that, in contrast to the SAI1 and SAI2 networks, BONSAI testbed provides a maximum speed of only 100 Mbits/s. Another comparison is taken from [18] (RBD). Here a 1 Gbits/s switch is used. Nonetheless, the specifications of the hosts used for the experiment far exceed the ones from BONSAI testbed. Finally, a tighter value is derived from [19] (Toronto) for migration between enterprise clouds of a VNF of Ubuntu 12.04 of 243.6 MB using a flavor comparable to F4.

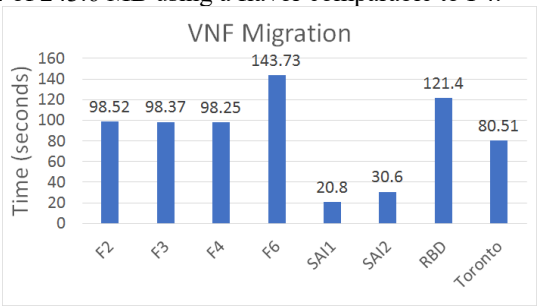

Fig. 3. VNF MIGRATION TIME

As we can see in the Fig.3, the time to complete the VNF migration, as we can infer from the results provided by the literature, greatly affects the performance of a network service. Despite the differences in throughput between the BONSAI testbed and the architectures proposed by the literature, such as the ones used on SAI1, SAI2 and Toronto, the results are comparable.

C. SENATUS overhead

Here we present the results from the experiments comparing the overhead introduced by SENATUS against the use of a script that exploits the Command Line Interface (CLI), which is the most common way of using Openstack without a service orchestrator. Fig 4 shows the time required for the communications for each API: Openstack and Gnocchi. On

Openstack, we have taken as reference the creation time of a VNF. Using the flavor F2 and CirrOS VNF of previous subsection, we measure the time between summoning the VNF creation and the instant when the VNF start running.

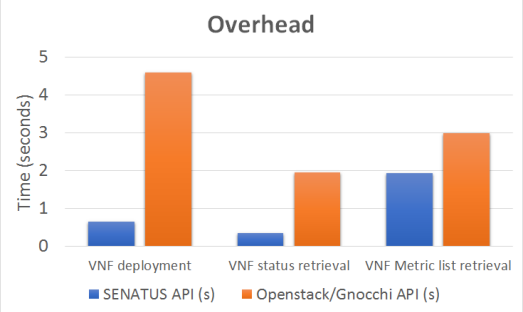

Fig. 4. SENATUS OVERHEAD

On the second experiment, we measure the time to recover information from Openstack, specifically, a request of the status of a VNF. Finally, we check the time required to recover a set of metrics from Gnocchi. The results above show that SENATUS is remarkably faster that the usage of the CLI for summoning actions and collect information. This is possible due to the direct usage of the Openstack and Gnocchi libraries, in contrast to the CLI methodology that must spend some time to convert the terminal input to API calls.

D. ONOS experiment

On this last set of experiments, we present some results from the SDN-related features of SENATUS. Experiment have been performed on the Mininet testbed.

Table 3. FLOW INSTALLATION TIME FOR DIFERENT SDN CONTROLLERS

Action	SENATUS+ ONOS (ms)	OpenDayLight (ms)	Ryu (ms)
Flow installation	1.52	1.5	0.3

In addition, as done with VNF migration experiment, we introduce some results from literature [19]. In particular, in Table 3 we compare the values of flow installation with another two commonly used SDN controllers: OpenDaylight [21] and Ryu[3]. It is observed that installing a flow from SENATUS using ONOS has similar performance than the provided by OpenDaylight, and both of them are much slower than Ryu. ONOS and OpenDaylight provides a full network operating system and run and support native network applications, while Ryu is based on a lighter approach as a python framework for SDN application development: thus, a faster response on flow installation was expected.

VI. CONCLUSIONS

In this work, we show a new service orchestrator called SENATUS. It provides service orchestration for network segments deploying Openstack as infrastructure manager and ONOS as SDN controller. SENATUS innovates with the provisioning of external modules that accomplishes the task of providing support for decision-taking algorithms for network planning. Furthermore, SENATUS in conjunction with ONOS and Openstack implements a rich variety of metrics that are available for collecting measurements of different performance of the system. In order to provide the full research ecosystem, SENATUS is presented together with two testbeds that have been deployed and configured as part of the architecture being studied.

[3] Weblink: https://osrg.github.io/ryu/

ACKNOWLEDGEMENTS

The work leading to these results has been supported by the European Community under grant agreement no. 761727 Metro-Haul project.

REFERENCES

[1] 5GPPP Architecture Working Group. Vision on software networks and 5g. January 2017.

[2] Cisco Visual networking Index. Forecast and methodology, 2016-2021, white paper. San Jose, CA, USA, 1, 2016.

[3] 5G PPP Architecture Working Group et al. View on 5g architecture. White Paper, July, 2016.

[4] 5GPPP Architecture Working Group. 5g empowering vertical industries. February 2016.

[5] 5GPPP Architecture Working Group. 5g ppp use cases and performance evaluation models. 2017.

[6] L. Peterson, et al. Central office re-architected as a data center. IEEE Communications Magazine, 54(10):96–101, October 2016.

[7] R. Vilalta, et al. Multi-tenant transport networks with sdn/nfv. In 2015 European Conference on Optical Communication (ECOC), pages 1–3, Sept 2015.

[8] R. Muñoz, et al. Sdn orchestration and virtualization of heterogeneous multi-domain and multilayer transport networks: The strauss approach. In 2015 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), pages 142–146, May 2015.

[9] A Israel, et al. Osm release three-a technical overview. White paper, ETSI, 2017.

[10] M. Mechtri, et al. Etso: End-to-end sfc orchestration framework. In 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), pages 903–904, May 2017.

[11] P. Berde et al., “ONOS: towards an open, distributed SDN OS”, ACM HotSDN pp. 1-6 (2014).

[12] Weblink: https://gnocchi.xyz/.

[13] S. Troia, et al. "Machine-Learning-Assisted Routing in SDN-based Optical Networks". Accepted in European Conference on Optical Communication (ECOC), Rome, Italy September 2018.

[14] Weblink: http://mininet.org/.

[15] A. Khalid, et al: A network animator for visualizing real-time packet flows in mininet. In Innovations in Clouds, Internet and Networks (ICIN), 2017 20th Conference on, pages 229–231. IEEE, 2017.

[16] A. Botta, et al, "A tool for the generation of realistic network workload for emerging networking scenarios", Computer Networks (Elsevier), 2012, Volume 56, Issue 15, pp 3531-3547.

[17] M. I. Biswas, et al. An analysis of live migration in openstack using high speed optical network. In 2016 SAI Computing Conference (SAI), pages 1267–1272, July 2016.

[18] W. Ding, et al. Construction and performance analysis of unified storage cloud platform based on openstack with ceph rbd. In 2018 IEEE 3rd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), pages 135–141. IEEE, 2018.

[19] D. Kargatzis, et al. Virtual machine migration in heterogeneous clouds: from openstack to vmware. In 2017 IEEE 38th Sarnoff Symposium, pages 1–6, Sept 2017.

[20] C. Metter, et al. Investigating the impact of network topology on the processing times of sdn controllers. In Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on, pages 1214–1219. IEEE, 2015.

[21] M. Jan, et al. "Opendaylight: Towards a model-driven sdn controller architecture", World of Wireless, Mobile and Multimedia Networks (WoWMoM), (2014).