SDN/NFV VNF Service Chaining

Dashmeet Anand, Hariharakumar Narasimhakumar, Rohit Kulkarni, Sarang Ninale, Levi Perigo, Dewang Gedia, and Rahil Gandotra

***Abstract*— Service Function Chaining (SFC) is a capability that links multiple network functions to deploy end-to-end network services. By virtualizing these network functions also known as Virtual Network Functions (VNFs), the dependency on traditional hardware can be removed, hence making it easier to deploy dynamic service chains over the cloud environment. Before implementing service chains over a large scale, it is necessary to understand the performance overhead created by each VNF owing to their varied characteristics. This research paper attempts to gain insights on the server and networking overhead encountered when a service chain is deployed on a cloud orchestration tool such as OpenStack. Specifically, this research will measure the CPU utilization, RAM usage and System Load of the server hosting OpenStack. Each VNF will be monitored for its varying performance parameters when subjected to different kinds of traffic. Our focus lies on acquiring performance parameters of the entire system for different service chains and compare throughput, latency, and VNF statistics of the virtual network. Insights obtained from this research can be used in the industry to achieve optimum performance of hardware and network resources while deploying service chains.**

***Keywords*— Cloud Orchestration, OpenStack, Service Function Chaining (SFC), Virtual Network Functions (VNFs).**

I. INTRODUCTION

Orchestration of network functionalities for growing networks demand greater flexibility than that offered by current traditional networks. The requirement for dynamicity in today's networks and data centers that house multi-tenant public/private infrastructures have resulted in the adoption of Virtual Network Functions (VNFs) [1]. Network Function Virtualization (NFV) is a network architecture concept that decouples the software of network functions from their hardware counterparts so that they can be hosted on commodity hardware (servers) to achieve an entirely virtualized

infrastructure [2]. Any network function software that can be used as a standalone entity in the NFV framework is known as a Virtual Network Function (VNF). As data center architectures are moving towards virtualization, VNF's have become more popular as they provide flexibility in design, reduce deployment time, and optimize network resources. This results in reduced Capital Expenditure (CAPEX) and Operational Expenditure (OPEX) compared to their traditional counterparts [3].

VNFs can provide standalone or coupled functionalities – for instance, a VNF could provide routing capabilities alongside firewall operations (such as Juniper VSRX VNF [4]), but a few VNF's in the market have limited capabilities such as Network Address Translation (NAT) provided by VyOS [5]. By chaining multiple such VNFs together, the network administrator can tailor the behavior of the network dynamically [20 - 26]. This process is known as Service Chaining or Service Function Chaining (SFC). SFC has therefore evolved as a capability that uses Software Defined Network (SDN) and Network Functions Virtualization (NFV) to construct a chain of network services (such as Firewall, NAT, and many more) [6].

This research makes use of OpenStack [7], an open source platform for cloud orchestration deployment. It offers capabilities to control large pools of compute, storage and networking functionalities. The different components of OpenStack that have been utilized in this research are glance (image module), Neutron (networking module), Nova (compute module) and Horizon (OpenStack dashboard). The OpenStack along with its compute node is used to host multiple VNFs and the network node is used to bind the VNFs together in a single service chain [8]. The neutron modules of the OpenStack also allow routing the traffic end to end within the OpenStack environment [9].

Even though the ability to connect these network services is well known, existing research lacks evaluation of the performance characteristics of chaining multiple VNFs. This research study aims to understand the various parameters that influence the underlying limited hardware resources in the deployment of these NFV/VNF Service chains. Our focus is to test the operation and performance of service chains and quantify the impact each VNF, and effectively the service chain has on the server resource utilization and network parameters.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

II. Related Work

Traditionally, network functionalities such as Network Address Translation (NAT), traffic filtering firewall, packet inspection or manipulation, and intrusion detection were implemented in the form of hardware devices called middleboxes. These devices offered new features, improved security, and improved performance [10].

With the highly dynamic network requirements that exist today, the deployment and operation of middleboxes come at the expense of increased capital and operational costs. To overcome this, Network Function Virtualization (NFV) was a scalable software solution that can virtualize the hardware middleboxes. The concept of NFV was introduced to replace middleboxes with commodity servers running software applications that replicate a network function (VNF) [11].

VNFs have an advantage over their physical counterparts (middleboxes) as they are easy to deploy, modify and upgrade.

[12] showcases that the placement of VNFs is a complex problem and decisions must be taken according to the desired objective (such as maximized data rate, low latency, etc.). Further investigation was suggested for obtaining optimized performance using efficient placement of VNFs in the service chain [12].

Since the advent of VNF based service chaining, research has helped maximize the performance of the service chain. Containers can be used to implement VNF service chains and introduce less kernel overhead into the system compared to virtual machines (VMs) [13]. However, the focus of this research is to evaluate the performance of VM based service chains using OpenStack cloud orchestrator.

Prior work lacked experimental results on the defining efficiency provided by the order of VNFs in a service chain and quantitative analysis of its throughput and performance characteristics. This research aims to fill this gap in the literature.

III. METHODOLOGY

A. *Overview*

The research intends to test the functioning of a service chain on a standalone server which houses OpenStack. Traffic entered the host server from the iPerf3 [14] client placed outside the host machine. The underlying operating system on the host machine forced the traffic into the SFC and finally pushed the traffic out of the interface connected to the iPerf3 server.

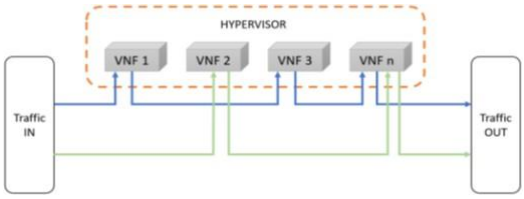
B. *Functionality:*

Fig. 1 shows the functionality of the complete infrastructure setup. “Traffic IN” serves as the client for traffic generator. VNF 1,2,3,…x are different instances that form the service chain deployed by OpenStack. Finally, “Traffic Out” is the server for the traffic generator. Traffic was sent from the client to the server.

In Fig. 1, traffic entered the server and was directed to the service chain via a routing policy that matches the source IP and incoming interface. Here, traffic entered VNF 1, which operated on the data and directed it to VNF 2, VNF 3, and so on to VNF

n. Traffic then left the service chain and consulted the routing table of the host machine to get to the traffic generator server.

Fig. 1. Functional Block Diagram.



C. The concept of Operations:

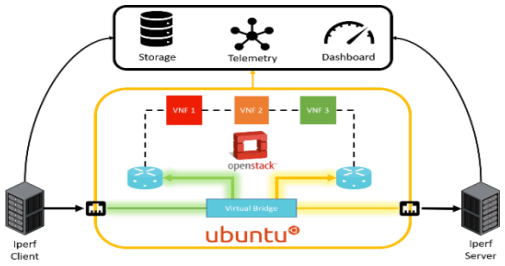


Fig. 2. Concept of Operation Bloack Diagram.

Fig. 2 shows an overview of the operational setup. The traffic entered the service chain inside the host machine, got processed by the chain, and reached the traffic sink. The complete environment was monitored and performance parameters were collected for monitoring and further analysis.

D. *Research Components:*

The research was developed in the following phases.

1) *Service Chain Implementation:* OpenStack on a host machine running Ubuntu operating system was used for the deployment of the VNFs. Different images of VNFs were gathered from the Open Source community, VyOS as well as vendors such as Cisco CSR and Juniper VSRX to test service chains.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

2) *Monitoring, Storage, and Dashboard:* Integration of different databases are used for collection and monitoring of parameters. Gnocchi [15] and OpenStack – Ceilometer [16] is used to collect metrics from the VNF's. Prometheus [17]– Node Exporter [18] is used for gathering server parameters and Influxdb is used to gather other parameter values. These data sources are then used by Grafana [19] – Graphical dashboard to display the stats.
3) *Traffic generator:* iPerf3 server and client on independent servers are deployed to push traffic across the host machine. iPerf3 is an open source utility and has multiple operations and flexibility in designing the traffic type.
4) *Traffic Flow/Networking:* The traffic from the iPerf3 server and client shall enter the host machine's operating system's networking components. Policy based routing is used to force the traffic from the interface into the OpenStack service chain. Each VNF is deployed on virtual networks created within OpenStack – Neutron.

Precautions were taken to ensure that service chains are tested in a consistent environment and that there is verification before the collected data is passed for any inference. Since the research was implemented on physical hardware, controlling variables in every test was a challenge. The following sections describe the key elements that verified the test results.

E. Testing Infrastructure Setup

The research kept the environment of the underlying system consistent across test cases. This test is to verify the status of the underlying infrastructure that will support the testing of the service chain. Before implementing the service chain and subjecting it to the network traffic, the server parameters were gathered.

Initial testing of the server was done by a simple Python program that gathered system resources. This baseline information is necessary to determine the effect a service chain would have on system resources. The program gathered information from server metrics which include: power state, temperature, RAM, memory, CPU type & utilization and interface status. It continues to test the connectivity of the source and destination (traffic generator ends), prints the OpenStack version and checks a basic operation by querying the image list from OpenStack - Glance.

F. Implementing Service Chain

This test ensured that the service chain is implemented successfully and that the traffic from the host server is routed via the service chain before it is sent out the exit interface.

Traffic entering the server interface needs to be forced into the service chain using policy-based routing, and then back out into the host servers routing plane. This test ensured that the host server does not route traffic at its networking layer i.e. it does not bypass the service chain altogether.

Permitting ICMP traffic inside the service chain allowed the traceroute application to gather the number of hops it passes through before exiting the host server. Comparing with the number of virtual networks deployed by Neutron, flow of traffic across each node in OpenStack service chain was confirmed.

G. Metric Collection

Variations in the performance parameters of different VNFs are monitored and collected for future analysis. VNF parameters are gathered using Ceilometer-Gnocchi, Ubuntu server parameters using Prometheus-Node Exporter, and network parameters from the output of iPerf and ping commands. A data visualization tool (Grafana) gathers up the data collected and represents it in a graphical utility for improved understanding of the service chain operation over time. The data is polled at regular intervals and written into Gnocchi and Prometheus databases. Host statistics and traffic generators output are validated across the data stored in the database to ensure data integrity. This was done periodically during each test case to ensure correct data was polled and stored in the server for references.

H. Reproducibility.

Finally, the data gathered from the tests needed to be re-verified. A secondary server with a similar configuration as the primary server is hosted in the test environment. The above- mentioned test cases were executed in the secondary server. The results obtained from the metric collection test case is compared with the information gathered from the primary server. If the results obtained were within five percent margin, then the test can be deemed successful.

Following sections contain the details of the hardware setup deployed in the lab. Four servers were deployed: two identical Host servers, one iPerf client and one iPerf server.

I. Hardware

1) *Host Servers:* The research uses two identical servers (Dell EMC Poweredge R430) for reproducibility of test cases. Enlisting few specifications of Host servers in table I.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

TABLE I. Host server specification

Feature	Specifications
Processor	Intel® Xeon® processor E5-2600 v4 product family
Memory	64GB
Storage	500GB SATA
Networking	4x1Gb Ethernet NIC's
Quantity	2

2) *Traffic Generator Servers:* Deployed over two Ubuntu Servers running iPerf3 software as traffic generators. Both the server and Client have 1Gb Ethernet NICs.

J. Software

Operating Systems: Ubuntu 16.04 LTS was the Operating Sytem for all servers.

1) Cloud Orchestration: OpenStack - Stable/Rocky version was deployed on Host machines for orchestrating VNF deployment. Gnocchi with Ceilometer was employed for monitoring VNFs. OpenStack installation was done using Devstack.

2) Traffic Generator: iPerf3 was selected as the traffic generator for measuring throughput for the network. The toolkit consists of a combined client/server program and customizable options for traffic type. iPerf3 software is relatively simpler and supports multiple parallel connections.

3) Graphical Dashboard: Grafana was deployed to provide a graphical dashboard for the collected metrics. The system integrated with multiple types of databases and OpenStack. It is an open source software and houses a customizable dashboard.

4) Database Source: Gnocchi was used for OpenStack VNF metric collection, Influxdb for storing data from program's output and Prometheus with the node-metric plugin for host server's stats collector.

This research intends to determine the most optimal combination of service chain based on key performance parameters such as CPU utilization, memory utilization, load average, throughput and latency across the chain as defined below:

1) *CPU utilization*: It is a measure of the number of tasks handled by the CPU in terms of processor cycles. Also, a task's run time is proportional to the CPU utilization of the host, thus directly affecting the performance. CPU utilization is an important parameter to monitor as it will show us how VNFs behave when subjected to traffic.

2) *Memory utilization*: The amount of memory allocated to processes to perform a set of tasks. We will be monitoring memory usage to determine what memory is being consumed by the VNFs to process the incoming traffic.

3) *Load average*: It represents the load on a system over various timestamps., for example, one minute and five minutes. It is calculated as the sum of number of processes being executed and the processes waiting for the CPU. This parameter will tell us the system behavior for longer durations.

4) *Throughput*: Throughput is the amount of actual data passing through a link. We'll measure throughput across the service chain and compare it with the capacity of the links.

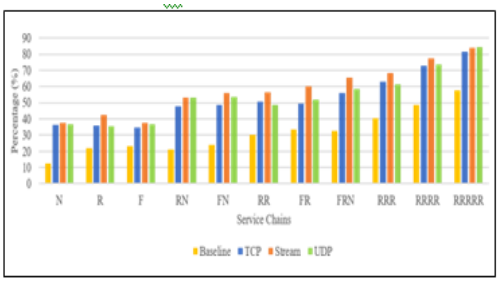
5) *Latency*: The time taken by a packet to traverse to the destination. We will evaluate latency as it essentially signifies the network performance.

IV. RESULTS

A. *Server CPU utilization:* It was observed that with each additional VNF (except VYOS NAT), the server CPU utilization percentage increases by 8-12 when no traffic is passed through the service chain. VYOS has negligible effect on the server CPU utilization , followed by CSR (router) and VSRX (firewall) in increasing order. Parallel TCP streams consume highest CPU on the server, followed by UDP, and finally TCP. The rate of increase in CPU utilization decreases with the addition of each router when traffic is passed through the service chain. The linear inclination of CPU Utilization is shown in graph I.

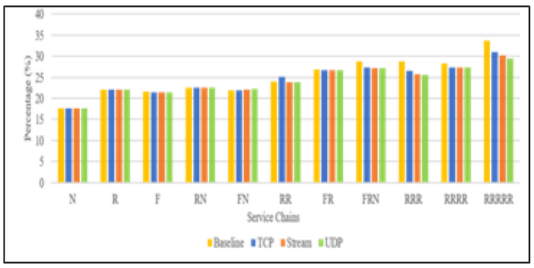
ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

GRAPH I : Server CPU Utilization.



B. Server Memory utilization (RAM): Increases with increase in the number of VNFs in the service chain with no discernible pattern. Passing traffic through the service chain has no effect on the memory utilization of the server and is unaffected by the nature of traffic as seen in graph II.

GRAPH II : Server Memory Utilization.

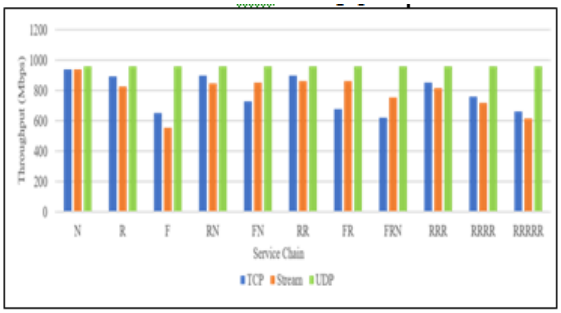


C. Latency: Increases with increase in the number of VNFs in the service chain. No relation observed between the type of VNF added and resulting latency. The Graph III shows the service chain and the respective latency of round-trip time in miliseconds observed across the network.

GRAPH III : Latency.

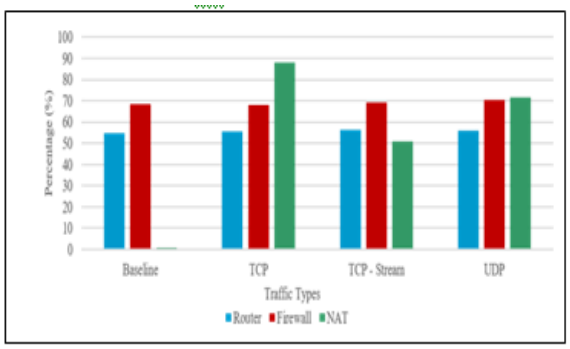
D. Throughput: Highest throughput is obtained for UDP traffic (and stays the same for every service chain), followed by TCP trafffic and finally parallel TCP streams as shown in graph IV. Throughput for all traffic types (except UDP) decreases with increase in the number of VNFs in the service chain. Throughput loss per individual VNF was highest in VSRX (firewall), followed by CSR (router), and then VYOS (NAT).

GRAPH IV : Throughput.



E. VNF CPU utilization: VSRX (firewall) consumes highest VNF CPU followed by CSR (router). For both the above VNFs, CPU utilization remains constant with each additional VNF and when traffic is passed through the service chain. VYOS (NAT) CPU utilization is negligible when no traffic is passed through the service chain but increases when subjected to traffic. Graph V shows the different VNF CPU Utilization across different traffic types.

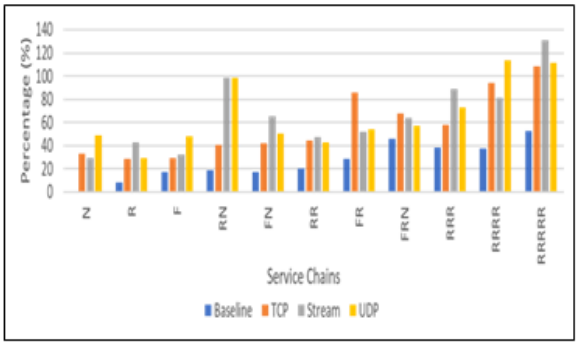
GRAPH V : VNF CPU Utilization.



F. Uncertainties: A few uncertainties were observed when parameters such as server CPU load and UDP datagram loss percentages were analyzed. There was no relation to the number of VNFs and the varying CPU load. No conclusion could be drawn from UDP datagram loss results obtained for each service chain. The graph VI shows the CPU load average calculated per minute from the server with respect to different service chain deployed and after subjecting to different test scenarios. Graph VII shows the UDP datagram loss percentage across different type of service chains deployed over the test environment. Both the graphs show inconclusive patterns.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

GRAPH VI : CPU Load.



GRAPH VII : UDP Datagram Loss.

V. CONCLUSION

The results are specific and dependent on the type of research's test environment. If the components of the test environment change, there may be some variation in the test results. However, the research analysis aims to provide patterns that can help in understanding and developing a rational behavior of the service chains with respect to the parameters mentioned below.

CPU Utilization, Memory Utilization and CPU Load of the server are directly affected by the type of VNFs, the number of VNFs deployed and the type of traffic passed through the service chain. All the above-mentioned parameters increase with an increase in the number of VNFs. Out of all the VNFs VSRX firewall requires the highest CPU followed by Cisco CSR. TCP parallel streams consume the most CPU amongst all the traffic types. Memory Utilization increases only with the addition of new VNF and remains constant thereafter. Throughput loss is the highest when passed through Juniper VSRX firewall. Additionally, lowest throughput is observed in the case of TCP parallel streams followed by UDP for most of the VNFs.

VI. FUTURE WORK

The research aims to help industry professionals to gain an insight into the performances of different Service Chains before deploying them onto the production network. Following modifications can be made to increase the scope of testing carried out in the research and make the tests suited for different production environments.

The designed testbed can be used to test various other VNFs and service chains that were not a part of this research. It can also be used to test service chains with different vendor-specific implementations of OpenStack.

The research sets an example that can be replicated on multi- node OpenStack services. The research architecture will also be used to test service chain in different network orchestrators other than OpenStack.

VII. REFERENCES

[1] R. Narayanan, V. Saxena, C. Tato, H. Nakamura, H. Wang, B. Lei, S. Rao, "Finding an Efficient Virtual Network Function Architecture for Next- Generation Telecommunications Infrastructure", [Online]. Available:https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/wp/wp-01273-finding-an-efficient-virtual-network- function-architecture.pdf. [Accessed: 02-Apr-2019]

[2] M. Mechtri, C. Ghribi, and D. Zeghlache, "A Scalable Algorithm for the Placement of Service Function Chains," IEEE Transactions on Network and Service Management, vol. 13, no. 3, pp. 533–546, 2016.

[3] F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba and O. C. M. B. Duarte, "Orchestrating Virtualized Network Functions," in IEEE Transactions on Network and Service Management, vol. 13, no. 4, pp. 725-739, Dec. 2016. doi: 10.1109/TNSM.2016.2569020

[4] Juniper Networks, "vSRX VIRTUAL FIREWALL". [Online]. Available:https://www.juniper.net/us/en/local/pdf/datasheets/1000489- en.pdf. [Accessed: 02-Apr-2019].

[5] "VyOS - an Open Source Linux-based Network OS," VyOS. [Online]. Available: https://vyos.io/. [Accessed: 03-Apr-2019].

[6] P. Quinn, T. Nadeau, and Ed, "Problem Statement for Service Function Chaining," RFC Editor, 01-Jan-1970. [Online]. Available: https://www.rfc-editor.org/info/rfc7498. [Accessed: 02-Apr-2019].

[7] O. Sefraoui, M. Aissaoui, and M. Eleuldj, "OpenStack: Toward an Open- source Solution for Cloud Computing," International Journal of Computer Applications, vol. 55, no. 3, pp. 38–42, 2012.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

[8] "OpenStack Compute (nova)" OpenStack Docs: OpenStack Compute (nova). [Online]. Available: https://docs.openstack.org/nova/rocky/. [Accessed: 02-Apr-2019].

[9] "Welcome to Neutron's documentation!" OpenStack Docs: Welcome to Neutron's documentation! [Online]. Available: https://docs.openstack.org/neutron/rocky/. [Accessed: 02-Apr-2019].

[10] B. Carpenter, "Middleboxes: Taxonomy and Issues", Ietf.org, 2002, [Online]. Available: https://tools.ietf.org/html/rfc3234. [Accessed: 02-Apr-2019]

[11] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On Orchestrating Virtual Network Functions in NFV", Arxiv.org, 2015. [Online]. Available: https://arxiv.org/pdf/1503.06377.pdf. [Accessed: 02-Apr-2019]

[12] S. Mehraghdam, M. Keller, H. Karl, "Specifying and placing chains of virtual network functions - IEEE Conference Publication", Doi.org, 2014. [Online]. Available: https://doi.org/10.1109/CloudNet.2014.6968961. [Accessed: 02-Apr-2019]

[13] S. Livi, Q. Jacquemart, D. L. Pacheco, G. Urvoy-Keller, "Container-Based Service Chaining: A Performance Perspective - IEEE Conference Publication", Doi.org, 2016. [Online]. Available: https://doi.org/10.1109/CloudNet.2016.51. [Accessed: 02-Apr-2019]

[14] V. GUEANT, "iPerf - The ultimate speed test tool for TCP, UDP and SCTPTest the limits of your network Internet neutrality test," iPerf.fr. [Online]. Available: https://iperf.fr/. [Accessed: 03-Apr-2019].

[15] "Gnocchi – Metric as a Service" Gnocchi – Metric as a Service - Gnocchi 4.2.1.dev96 documentation. [Online]. Available: https://gnocchi.xyz/. [Accessed: 03-Apr-2019].

[16] "Welcome to Ceilometer's documentation!¶," OpenStack Docs: Welcome to Ceilometer's documentation! [Online]. Available: https://docs.openstack.org/ceilometer/latest/. [Accessed: 03-Apr-2019].

[17] Prometheus, "From metrics to insight," Prometheus Blog. [Online]. Available: https://prometheus.io/. [Accessed: 03-Apr-2019].

[18] Prometheus, "Monitoring Linux host metrics with the Node Exporter," Prometheus Blog. [Online]. Available: https://prometheus.io/docs/guides/node-exporter/. [Accessed: 03-Apr-2019].

[19] "Grafana - The open platform for analytics and monitoring," Grafana Labs. [Online]. Available: https://grafana.com/. [Accessed: 03-Apr-2019].

[20] D. Gedia, and L. Perigo, "NetO-App: A Network Orchestration Application for Centralized Network Management in Small Business Networks," CSITY, DTMN, NWCOM, SIGPRO – 2018, pp. 61–72.

[21] D. Gedia, and L. Perigo, "A Centralized Network Management Application for Academia and Small Business Networks," IT in Industry (ITII), vol. 6, no.3, Aug. 2018.

[22] D. Gedia, and L. Perigo, "Latency-aware, Static, and Dynamic Decision-Tree Placement Algorithm for Containerized SDN-VNF in OpenFlow Architectures", in 2019 IEEE Conference on Network Function Virtualization and Software Defined Networks. IEEE, Nov, 2019, Dallas, TX.

[23] R. Gandotra, and L. Perigo, "SDNMA: A Software-defined, Dynamic Network Manipulation Application to Enhance BGP Functionality" in 20th IEEE International Conference on High Performance Computing and Communications (HPCC-2018), June, 2018.

[24] D. Gedia, and L. Perigo, "Performance Evaluation of SDN-VNF in Virtual Machine and Container", in 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN). IEEE, Nov, 2018, Verona, Italy.

[25] A. Jain et al., "Trend-Based Networking Driven by Big Data Telemetry for SDN and Traditional Networks," International Journal of Next-Generation Networks (IJNGN), vol. 11, no. 1, 2019.

[26] M. Jain, S. Suneja, S. Srivatsa, V. Ananthasubramanian, Y. Maramraj, L. Perigo, R. Gandotra, and D. Gedia, "Intent-Based, Voice-Assisted, Self-Healing SDN Framework," Journal of Network Communications and Emerging Technologies (JNCET), Vol. 10, Issue 2, Feb., 2020.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731