Improving Orchestration Service Using gRPC API and P4-Enabled SDN Switch in Cloud Computing Platform: An OpenStack Case

Hari Krishna S. M, Rinki Sharma

***Abstract*— Cloud computing platforms like OpenStack use orchestration as a crucial component in the deployment and administration of cloud services. For the deployment and management of virtualized resources, these platforms mainly rely on orchestration techniques. Cloud orchestration services like Service Function Chaining (SFC) have benefited from lower costs and increased scalability for cloud computing platforms thanks to the introduction of technologies like Software-Defined Networking (SDN) and Network Function Virtualization (NFV). The use of gRPC APIs and an SDN switch with P4 support is described in this research as an innovative method for enhancing SFC orchestration service in OpenStack computing platforms. The proposed approach uses a P4 enabled SDN switch and gRPC APIs to enhance network performance in data plane communication with the underlying infrastructure and performance in terms of service creation time between the SFC orchestration services. Our evaluation's findings demonstrate that this strategy greatly boosts the orchestration service's performance, making it more appropriate for usage in extensive cloud-based deployments. Additionally, the possibility of combining gRPC APIs and P4 switch to improve the efficiency and flexibility of orchestration service in cloud computing platforms is demonstrated by the performance analysis of the proposed approach on numerous platforms across diverse use cases.**

***Index Terms*— GRPC, Orchestration, OpenStack, P4, REST.**

I. INTRODUCTION

Cloud computing platforms are utilized for offering infrastructure-as-a-service (IaaS) and platform-as-a-service (PaaS) capabilities. These platforms empower cloud service providers to efficiently create and oversee virtual machine (VM) instances, storage, and networking resources within a cloud environment [1].

Through the provision of Application Programming Interfaces (APIs) and Graphical User Interfaces (GUIs), users are enabled to automate and control the process of creating and managing cloud resources. The Software-Defined Networking (SDN) architecture separates the control plane and data plane in a network. SDN aims to deliver a highly flexible software-driven network that supports dynamic and high-bandwidth applications, such as cloud computing, while ensuring superior performance and minimal overhead [2]. Network functions virtualization (NFV) enhances networking flexibility and agility by enabling the implementation of network functions in software, which can run on standard servers instead of relying on specialized hardware [3]. The combination of cloud computing platforms and technologies like SDN and NFV enables the establishment of more adaptable and agile networking in modern data centers and cloud environments. This empowers organizations to create and manage their networking resources swiftly and easily.

A. SFC Orchestration on SDN and NFV Platform

The effective deployment of infrastructure or a platform as a service in a cloud computing environment like OpenStack [4] relies heavily on the concept of Service Function Chaining (SFC) orchestration. SFC orchestration utilizes the combined power of SDN and NFV technologies to construct and manage intricate networking and computing environments [5]. NFV simplifies the deployment and configuration of instances by virtualizing network functions. On the other hand, SDN, a networking architecture, decouples the control and data plane, enabling the implementation of complex SFC. Through well-defined SDN APIs, the SDN controller acts as an abstraction layer between the control and data plane. The utilization of both SDN and NFV technologies contributes to enhancing the flexibility and scalability of SFC orchestration, enabling more efficient management of resources.

B. Components of SFC orchestration in OpenStack

Within the context of SFC orchestration in OpenStack, two crucial components are Open vSwitch (OVS) [6] and a REST (Representational State Transfer) API [7]. To execute the intended SFC, OVS is crucial in the creation and management of virtual switches. On the other hand, the REST API enables programmatic access to the functionality of the OVS-based network while facilitating communication between the key orchestration components. One of the most effective OpenFlow implementations is OVS, which is well known for this. In many different scenarios, including servers, hypervisors, and containers, it is widely used to provide virtual networking. OVS performs the function of a virtual switch for the installation of virtual networking in

Manuscript received February 22, 2023; revised July 19, 2023.

Hari Krishna S M is an Assistant Professor and Research Scholar, Department of Computer Science and Engineering, M S Ramaiah University of Applied Sciences, Bengaluru, India (corresponding author, phone: +918553935263, E-Mail: hk8892316082@gmail.com).

Rinki Sharma is a Professor and Head of the Department of Computer Science and Engineering, M S Ramaiah University of Applied Sciences, Bengaluru, India; (E-Mail: rinki.cs.et@msruas.ac.in).

cloud computing environments like OpenStack. By utilizing the OVS and REST APIs within OpenStack's Neutron networking service, users can create VM instances and other resources while enabling seamless communication between these components [8].

C. Advancing OpenFlow to P4

Due to its drawbacks, such as fixed header fields and the requirement to implement a newer version of the protocol, the OpenFlow protocol has drawn criticism. Additionally, the number of research contributions aimed at enhancing OpenFlow has decreased because of the advent of the Domain Specific Language (DSL) known as Programming Protocol-Independent Packet Processors (P4) [9]. P4 is becoming more popular as a competitive substitute, especially in the field of programmable data planes [10]. Through capabilities like customization, stateful packet processing, high programmability, and an efficient method for introducing new protocols from SDN controllers to switches [11], it provides more flexibility. The growing popularity of P4 signifies its potential to address the shortcomings of OpenFlow and provide advanced capabilities for network programming.

D. Contributions of the Paper

The performance of REST API and OVS in OpenStack cloud environments has raised concerns, especially when deploying services like IaaS. However, the evolution of networking components and APIs within OpenStack indicates that integrating P4-based SDN solutions and gRPC APIs could significantly enhance the performance of cloud-based connectivity services. P4 offers greater programmability and flexibility compared to OVS, enabling more efficient and advanced network management. It can be leveraged to implement advanced networking features and achieve high-speed packet processing. On the other hand, gRPC provides superior performance and efficient data interchange, making it particularly valuable for building high-performance, low-latency services.

The primary focus of this paper is to address the following objectives:

1. Enhancing the network performance of the data plane in SDN-enabled cloud computing platforms, such as OpenStack, by utilizing a P4 software switch.

2. Integrating gRPC API with the P4-based solution to improve the throughput of SFC orchestration, specifically reducing service creation time and enabling faster orchestration.

3. Developing SFC use cases and implementing SFC orchestration platforms using different combinations, including REST API with OVS, gRPC API with P4, REST API with P4, and gRPC API with OVS.

This paper presents a performance analysis of an SFC orchestration strategy developed using gRPC API and a P4-enabled software switch within the OpenStack cloud environment. To the best of our knowledge, this is the first attempt to quantify the performance gains achieved through SFC orchestration using gRPC API with P4-based SDN switch solutions in OpenStack. The analysis aims to explore the potential of gRPC API and P4 switch integration within OpenStack, paving the way for the platform to evolve into an integrated solution for NFV deployments at a production level. This integration would contribute to achieving a fully-fledged virtualized infrastructure management component.

E. Organization of the Paper

The rest of the paper is structured as follows: Section II presents the related work, a brief background of technologies and followed by related work with identified research gaps. Section III describes the design of the proposed orchestration strategy using gRPC API and P4 switch, including the procedure or algorithm used to implement them. Section IV details the deployment scenarios and subsequently covers the simulation results and comparative analysis for orchestration time and network performance. Section V concludes the paper by summarizing the research findings and presenting proposed future work.

II. Background And Related Work

This section gives a brief explanation of the underlying technologies before delving into a thorough analysis of relevant research on the topic, as well as prior studies that have addressed it. It also identifies any research gaps that need to be filled.

A. Background

Open vSwitch (OVS): OVS is an open-source software switch that is adaptable and scalable and may be utilized in a variety of settings. Numerous technologies and protocols are supported, including OpenFlow, which makes SDN possible. OVS is a virtual switching system that has grown in popularity in cloud environments and is widely used in data centers across the world [12].

P4: P4 is a programming language used to define packet processing in network devices. Its primary objective is to offer a high-level language for flexible and programmable packet processing pipelines. P4 has gained traction in programmable networking and is utilized in diverse environments such as data centers, service provider networks, and enterprise networks [13].

OpenStack: OpenStack is an open-source platform for building and managing cloud computing infrastructure. Comprising several components, it provides IaaS capabilities, including virtualization, storage, and networking functionalities. OpenStack has gained significant popularity and is adopted by various organizations, including large enterprises, service providers, and government agencies [14].

P4 Switch: P4 switches are network switches that can be programmed using the P4 programming language. They offer greater flexibility and dynamism compared to traditional switches. P4 switches are widely employed in

different environments, including data centers, service provider networks, and enterprise networks. They play a crucial role in SDN and are often used alongside SDN controllers to create highly programmable and flexible networks [15].

gRPC: gRPC is a modern, high-performance Remote Procedure Call framework that is open-source and platform-agnostic. It combines efficient inter-process communication with a simple, elegant design. The gRPC API enables communication between clients and servers using HTTP/2 as the transport protocol. Protocol buffers, a high-performance serialization mechanism, serve as the interface definition language (IDL) for defining services and data structures exchanged between the client and server [16].

B. Related Work

The discussion begins by focusing on the performance evaluation of network virtualization in cloud computing environments utilizing the OpenStack platform. Network virtualization enables the creation of virtual network infrastructure on top of a physical network, allowing for isolated and shared virtual networks for multiple tenants or users in the cloud. In a cloud computing system powered by OpenStack, the performance evaluation of network virtualization scenarios is presented in the research article [17]. The study evaluates alternative virtual switch implementations and looks at how different network topologies affect performance.

In a different study [18], the authors do experiments to evaluate the effectiveness of OpenStack's virtual networking, specifically for applications involving NFV. According to the findings, virtual networking in OpenStack demonstrates positive performance traits, such as reduced latencies and high throughput. The study also emphasizes how virtual networking in OpenStack results in efficient use of network resources and lessened network congestion.

Additionally, researchers [19] contend that incorporating native SDN into OpenStack can improve the platform's networking capabilities. Native SDN integration allows advanced networking capabilities like load balancing and Quality of Service (QoS) and promotes more effective use of network resources. The authors assess the performance of OpenStack with and without native SDN integration through their evaluation, finding that native SDN integration results in better network performance and less congestion. Additionally, they discovered that native SDN integration facilitates network management and configuration by offering more precise control over network behavior.

The evaluation of network performance improvement methods for cloud-native network functions (CNFs) is the subject of another paper [20]. CNFs, or cloud network functions, are software-based network services that are implemented in cloud settings and include services like firewalls, load balancers, and VPNs. The study investigates several methods, including making use of multiple processors (cores) for parallel processing, high-speed networking interfaces, and offloading and batching to cut down on packet processing overhead. The results show that the performance of CNF is greatly enhanced by several cores and fast networking interfaces. Depending on the workload and system design, offloading and batching strategies also show performance advantages. The research provides several ways to improve CNF network performance, with each method's success depending on certain conditions and limitations.

Another study [21] investigates how using a high-performance cloud infrastructure affects NFV performance. The authors run tests to gauge NFV performance on a powerful cloud platform designed for performance-critical applications. The study contrasts the performance of NFV on a high-performance cloud system with that of a regular cloud system and finds that the high-performance cloud environment exhibits noticeably increased performance. The results demonstrate that resource-intensive applications or those needing low latencies notably benefit from the high-performance cloud infrastructure. According to the research, using a high-performance cloud infrastructure can greatly improve NFV performance, especially for applications that require performance.

The survey report [11] gives a thorough summary of the OpenFlow protocol and how SDN has progressed in favor of more adaptable and programmable techniques like P4. The authors examine numerous NFV, data center networking, and IoT P4 applications and use cases within SDN. The report summarizes the body of literature and provides insightful analyses of the difficulties and possibilities for further study. The authors also execute tests to assess the effectiveness and performance of their suggested solution, contrasting it with existing NFV implementation strategies. For scholars and practitioners interested in the transition from OpenFlow to P4 in SDN, this paper is a great resource.

The authors of the paper [22] talk about the drivers for implementing hardware acceleration in NFV, highlighting the demand for enhanced performance and scalability. To offload packet processing work from the CPU, they discuss their architecture and implementation, which makes use of hardware acceleration technologies including FPGAs and NPUs. Another investigation [23] contrasts several orchestration techniques applied to the OpenStack cloud computing infrastructure. The goal of the paper is to shed light on the advantages and disadvantages of various orchestration techniques, allowing readers to assess how well they work in diverse environments or use cases.

An exploratory study assessing the design quality of REST APIs in cloud computing is presented in the research on cloud computing APIs in [24]. The authors evaluate the design and point out any potential problems or areas that may be improved.

Recently, [25] gave a thorough analysis of NFV and SFC frameworks. The authors talk about the history and inspiration behind NFV and SFC, describe the needs and goals of these frameworks, and give an overview of the current implementations, which include both closed-source and open-source options. They highlight the utilization of containerization and SDN in NFV and SFC, as well as well-known open-source systems like OpenStack and ONOS. The paper concludes by identifying open research challenges in the field, such as better integration with cloud

and edge computing and the development of efficient resource allocation algorithms.

C. Summary and Research Gaps in Existing Work

In the context of orchestration in cloud computing, there are several research gaps that can be addressed:

1. P4-enabled SDN switch integration in cloud computing platforms: Additional study is required to examine the integration of P4-enabled SDN switches, such as P4-based programmable data planes, with cloud computing platforms like OpenStack. By utilizing P4's skills for controlling and processing network traffic, this research may concentrate on improving network performance, flexibility, and programmability.
2. Increasing REST API performance: REST APIs are frequently used in cloud environments, but there is little study on how to do so. Future research can concentrate on increasing the throughput and response time of REST APIs while considering variables like network latency, scalability, and load balancing strategies. This study has the potential to boost the general effectiveness and responsiveness of cloud-based services.
3. Exploration of alternatives to REST APIs: Research may also investigate the viability of alternatives to REST APIs as a replacement or addition in cloud systems. Emerging technologies like gRPC, which offer better performance and effective data transmission, might be assessed as part of this exploration. Comparing the functionality and efficiency of various API solutions might shed light on how well-suited they are to various cloud use cases.
4. Addressing orchestration performance: The evaluation points out a research hole about improving orchestration performance, particularly about cutting down on orchestration time. Future research can concentrate on creating tactics and algorithms to streamline the orchestration procedure, cutting down on the amount of time needed for service creation and deployment. This study can help increase cloud orchestration performance overall and resource management efficiency.

The capabilities, performance, and scalability of REST APIs and OVS in the OpenStack ecosystem can be improved by filling in these research gaps. In the end, this will help meet the changing demands of cloud computing by improving administration and orchestration of cloud infrastructure.

III. Design And Implementation

A. Overview

Utilizing the gRPC API and a P4-based software switch, the architecture depicted in Fig. 1 shows a high-level design for implementing orchestration operations in the OpenStack cloud computing platform.

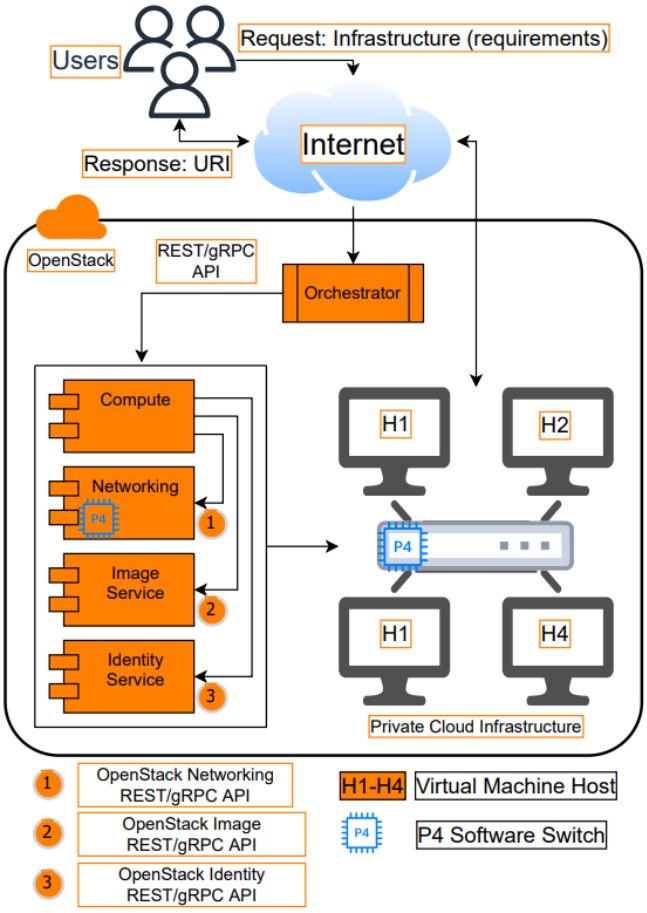

Fig. 1. SFC Orchestration in SDN and NFV Platform

The OpenStack cloud computing platform serves as the cornerstone of this architecture, offering the required services and infrastructure for managing and running cloud environments. Two essential elements, the gRPC API and a P4-based software switch are integrated into this platform to enable orchestration operations. As the communication interface that allows for interaction and data exchange between various system components, the gRPC API plays a crucial role. It makes RPC between the various orchestration entities effective and high performing. Another crucial component of this architecture is the P4-based software switch, which controls the virtualized networking features in the OpenStack environment. To enable advanced networking, it makes use of the adaptability and programmability provided by P4, a domain-specific language for packet processing.

By enhancing the orchestration processes within the OpenStack cloud computing platform, the gRPC API and the P4-based software switch enable greater performance, flexibility, and programmability in managing virtualized networks. To improve the orchestration capabilities in the cloud environment, this architecture offers a combination of the advantages of gRPC and P4 technologies.

When a cloud customer uploads their VM requirements via the OpenStack dashboard, the process of deploying VMs in an OpenStack environment gets started. The orchestrator component, which oversees the deployment process, then receives these requirements. The compute, network, image, and identity services, as well as other services offered by the OpenStack platform, are all in communication with the

orchestrator. Nova is the name of the main part of the compute service. The orchestrator uses APIs made available by Nova to get the data it needs from other services and install the virtual machines on the cloud. To obtain network information, the Nova component communicates with the neutron-server component of the networking service. Similarly, it retrieves image information from the glance-repository component of the image service. The orchestrator also requires authorization from the identity service component called Keystone to access and authenticate each service. Each of these services within OpenStack has its own set of existing REST APIs and gRPC APIs, which are used for inter-service communication. These APIs enable seamless communication between the services and facilitate the deployment of the virtual infrastructure within the cloud platform. Once all the requirements are met and the necessary information is obtained, OpenStack proceeds to deploy the virtual infrastructure in the cloud platform based on the user's specifications.

Fig. 2 illustrates a simple scenario depicting a deployed single tenant virtual infrastructure with three VM Hosts. In this scenario, two of the Hosts are implemented as VM Hosts within the OpenStack cloud environment and are interconnected. The third Host, referred to as the External Host, is implemented as another VM located outside the OpenStack cloud and is connected to the rest of the infrastructure via a local area network (LAN).

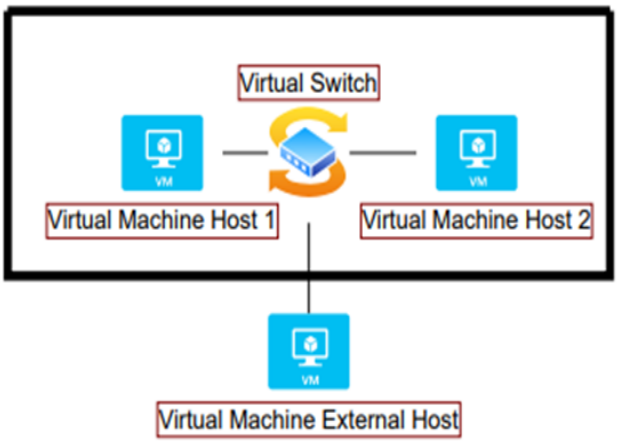

Fig. 2. Single Tenet Virtual Infrastructure Connected using Virtual Switch

B. *Virtual Network Infrastructure in OpenStack*

OpenStack provides physical nodes and network nodes to manage computing and networking resources. These can be achieved by either web-based user interface or flexible API designed based REST architecture. Fig. 3 illustrates a typical deployment of nodes in OpenStack cloud. The components are as follows:

i. Controller node: It is responsible for managing the OpenStack components in the cloud platform.
ii. Network node: It is used to host networking services such as internal connectivity for VM and external connectivity for connecting to external network.
iii. Compute nodes: One of the core components of OpenStack to execute the VMs.
iv. Storage nodes: It is used to store VM images and their related data.
v. Management network: Controller nodes manage OpenStack cloud services running in different nodes.
vi. Instance/Data network: It is used for connecting the network and the computer nodes for deploying virtual tenant networks with internal traffic and VM connectivity to the cloud networking services running in the network node.
vii. External network: It is used for enabling connectivity outside the internal network.

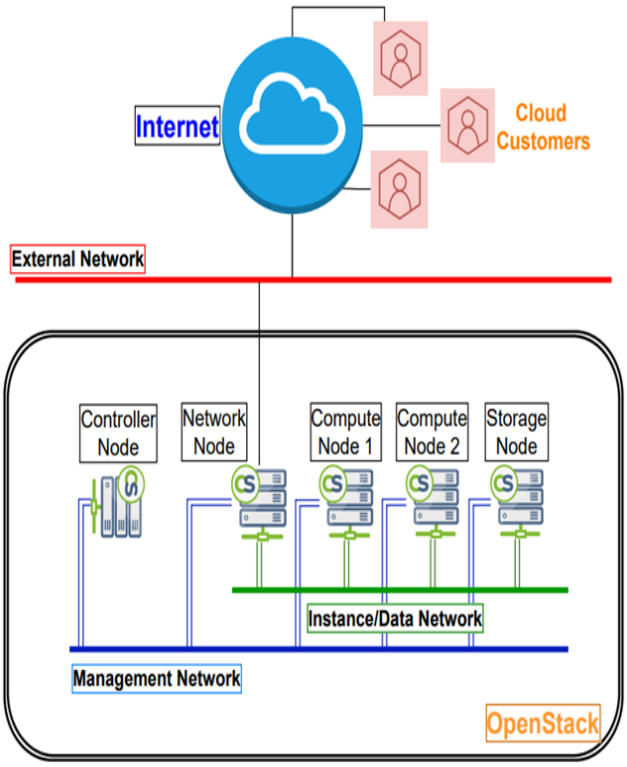

Fig. 3. Deployment of Nodes in OpenStack

The Neutron component is essential to network management in the OpenStack context. Neutron is made to provide users and administrators with a versatile interface for managing virtual networks. The hub of the Neutron component is the Neutron server, which is installed on the controller node. It supports the deployment of virtual network infrastructure in a dispersed environment and maintains network information. The management of multi-tenant networks across numerous compute nodes is made possible by Neutron, enabling seamless communication between VMs inside the architecture. Neutron integrates with SDN technologies to efficiently deliver and control network services. By offering cutting-edge networking services and features, SDN expands Neutron's capabilities. In the OpenStack environment, Neutron and SDN allow effective and scalable networking services.

C. *Modification to Neutron using P4 Switch*

The cloud user uses the OpenStack dashboard to create VMs, networks, and subnetworks in the OpenStack environment. The user can define the desired network and subnet settings, choose from the available images (Operating Systems), and describe the desired VM setup using the dashboard. The OpenStack component Neutron

oversees conducting network-related operations. Neutron generates a port on the designated subnet when constructing a VM and connects the VM to that port. The network's DHCP service then gives the VM a fixed IP address. Some VM instances require external users to be able to access them. This is accomplished by giving the VM a floating IP address that permits external access. The network node is set up with VM-specific forwarding rules to facilitate this. The OpenFlow protocol is then used to push these rules to the software switch, in this case the OVS.

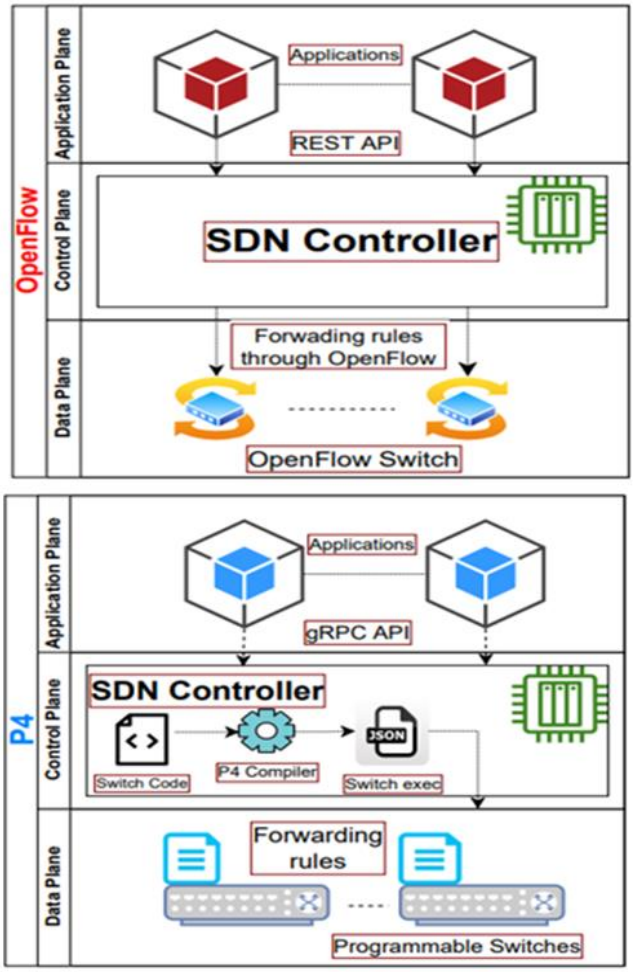

Fig. 4. OpenFlow based Switch and Modified P4 Switch.

Fig. 4 shows how OpenFlow and a modified P4 method can be combined in an SDN context. Although OpenFlow allows for the dynamic addition of flow rules for network traffic, its data plane capabilities are fixed. In contrast, the P4 language allows for programmability of the SDN's data plane, enabling customization of data plane protocols.

The proposed framework leverages a P4-enabled SDN switch, which offers programmability to the data plane. It exposes this programmability to the application plane through gRPC APIs. This allows external applications to interact with the dedicated data plane module via the gRPC API provided by the SDN controller. It's worth mentioning that the existing OVS, which is an OpenFlow-based switch, has an API based on the REST architecture.

However, the proposed modification involves integrating P4 switches with external applications, enabling them to interact with the data plane module using the gRPC API provided by the SDN controller.

D. Integration and Implementation

To enable SFC orchestration using gRPC API and P4 switch in the OpenStack cloud computing platform, the following algorithm outlines the necessary configuration steps and workarounds for successful integration:

Step 1: Deploy the OpenStack infrastructure and its components (Nova, Neutron, Keystone, Glance) in a virtual environment.

Step 2: Configure the P4 switch by mapping IP addresses, subnet masks, and default gateway settings to establish connectivity with the deployed OpenStack servers.

Step 3: Install and configure the OVS with the appropriate IP address and port settings to connect to the P4 switch.

Step 4: Install the gRPC library on the OpenStack server to enable gRPC communication.

Step 5: Create gRPC services and configure the gRPC server by defining the service address, service port, and gRPC service proto file. This file generates the necessary support files for the gRPC server and gRPC client.

Step 6: Create a gRPC client that can utilize the gRPC services to create service functions with the required parameters. Import the gRPC client library generated in step 5 for this purpose.

Step 7: Create a REST client that uses the existing OpenStack native REST APIs to create service functions with the necessary parameters.

Step 8: Create VMs using both the gRPC client (step 6) and the REST client (step 7) on the OpenStack environment. Test the connectivity between the VMs to ensure proper integration.

Step 9: Deploy the SFC with the desired virtual network topologies and configurations, leveraging the integrated gRPC API and P4 switch.

Step 10: Monitor the performance of the P4 switch with OVS and compare it with the performance of the gRPC-based orchestration services and the REST-based orchestration services in OpenStack.

By following these steps, the gRPC API and P4 switch can be effectively integrated into the OpenStack environment, providing enhanced functionality and performance for SFC orchestration. It is crucial to carefully configure the integration, considering compatibility and ensuring that the right settings are applied to achieve optimal results within the OpenStack cloud computing platform.

IV. Performance Evaluation

The performance evaluation of the implemented SFC

orchestration was conducted to assess two key aspects: overall service orchestration time and network performance with respect to the P4 switch. Additionally, a comparison was made between the performance of the existing system and the proposed solution.

1. Overall Service Orchestration Time: The length of time required for the service functions to be orchestrated within the SFC was measured and contrasted between the current system and the suggested solution. This comprises the time needed for service function configuration, deployment, and activation. Using the gRPC API and P4 switch, the suggested method seeks to optimize and shorten the total orchestration time.

2. Network Performance: Throughput, latency, and packet loss were some of the metrics used to assess the network's performance. To ascertain whether the integration of the gRPC API and P4 switch had any effect on network performance, the performance of the proposed solution was contrasted with that of the old system. This assessment aids in determining how well the suggested solution manages network traffic and provides dependable connectivity.

3. Comparison of Performance: The effectiveness of the current system and the suggested solution were thoroughly compared. Analysis of variables such overall service orchestration time, network performance measurements. Comparing these elements makes it feasible to spot any performance enhancements or downsides made by the suggested solution.

The performance evaluation sheds light on the advantages and disadvantages of the suggested solution in comparison to the current system. It helps assess whether faster service orchestration and better network performance will result from the combination of the gRPC API and P4 switch in the OpenStack system. These findings aid in evaluating the viability and efficacy of the suggested approach for SFC orchestration on the OpenStack cloud computing platform.

A. Test Bed: Prerequisites and Simulation Parameters

The proposed orchestration model for the OpenStack cloud computing server is tested using the following parameters:

- Server image: Ubuntu 20 server
- Server image size: 1.45 GB
- RAM: 512 MB allocated per each server image
- Number of Virtual CPU (VCPU): 4
- OpenStack version: Wallaby
- OpenFlow enabled switch: OVS
- P4 software switch model: BMv2
- Network performance tool: iperf2

The system is deployed on a virtual machine (VM) with Ubuntu 20 operating system, which has 12 GB RAM and 100 GB storage. The server image used is Ubuntu 20 server with a size of 1.45 GB. Each server image is allocated 512 MB of RAM and has 4 virtual CPUs (VCPU) available.

The OpenStack version used for the deployment and testing is Wallaby. The P4 software switch model chosen is BMv2, which provides programmability to the SDN's data plane and enables customization of data plane protocols.

For analyzing the network performance, the iperf2 tool is utilized. This tool allows measuring the throughput and performance of the network.

By conducting the testing and analysis with these parameters, the proposed orchestration model can be evaluated for its performance and functionality in the OpenStack cloud computing environment.

B. Results: OVS and P4 Switch

The network performance of the OVS and P4 switch was assessed by conducting experiments using the iperf2 tool. The experiments focused on sending UDP packets and evaluating the results for different scenarios involving varying bandwidth allocation and session time. The findings from these experiments are presented in Table I to Table IV.

The findings of the network performance analysis performed on the OVS switch are presented in Table I: Network Performance on OVS (UDP Packets, Varying Bandwidth Allocation). The studies comprised delivering UDP packets with changing bandwidth allotments while maintaining a consistent session length.

The outcomes of the network performance assessment on the P4 switch are shown in Table II: Network Performance on P4 Switch (UDP Packets, Varying Bandwidth Allocation). Similar to Table I, UDP packets were transmitted with varied amounts of bandwidth while keeping the session time constant.

The results of the network performance evaluation carried out on the OVS switch are presented in Table III: Network Performance on OVS (UDP Packets, Varying Session Time). This time, the studies centered on transmitting UDP packets while maintaining a steady bandwidth allotment.

The outcomes of the network performance analysis on the P4 switch are shown in Table IV: Network Performance on P4 Switch (UDP Packets, Varying Session Time). The tests entailed delivering UDP packets with various session lengths while keeping the given bandwidth constant.

These tables present valuable insights into the network performance of both the OVS and P4 switch under different scenarios, allowing for a comparison between the existing system and the proposed solution.

TABLE I
NETWORK PERFORMANCE OF OVS WITH VARYING BANDWIDTH

Time Allotted (Seconds)	Allotted Bandwidth (Mbps)	Bandwidth (Mbps)	Transfer Data (GB)	Jitter (ms)	Datagrams Sent (bytes)	Datagrams Lost (bytes)	Packet Loss (Percentage)	PDR (Packet Delivery Ratio)
300	10	9.87	0.346	0.08	255102	2214	0.867888139	0.991321119
300	20	19.5	0.681	0.114	510206	12415	2.433330851	0.975666691
300	30	28.4	0.991	0.04	765307	41298	5.396265812	0.946037342
300	40	37.4	1.309	0.05	1020408	67017	6.567667051	0.934323329
300	50	46.5	1.62	0.037	1275511	89398	7.008798827	0.929912012
300	60	55.68	1.941	0.09	1530612	111159	7.262415295	0.927375847
300	70	64.79	2.259	0.08	1785714	134056	7.507159601	0.924928404
300	80	73.91	2.576	0.022	2040816	156953	7.690717831	0.923092822
300	90	83.03	2.894	0.106	2295918	179850	7.833485342	0.921665147
300	100	92.15	3.211	0.086	2551020	202747	7.947699352	0.920523006

TABLE II
NETWORK PERFORMANCE OF P4 ENABLED SWITCH WITH VARYING BANDWIDTH

Time Allotted (Seconds)	Allotted Bandwidth (Mbps)	Bandwidth (Mbps)	Transfer Data (GB)	Jitter (ms)	Datagrams Sent (bytes)	Datagrams Lost (bytes)	Packet Loss (Percentage)	PDR (Packet Delivery Ratio)
300	10	9.89	0.345	0.088	255103	2720	1.066235991	0.98933764
300	20	19.7	0.688	0.097	510205	7037	1.379249517	0.986207505
300	30	29.7	1.04	0.083	765294	7050	0.921214592	0.990787854
300	40	40	1.4	0.082	1020409	961	0.094177923	0.999058221
300	50	50	1.75	0.036	1275511	247	0.019364788	0.999806352
300	60	60	2.101	0.041	1530610.4	232	0.015157352	0.999848426
300	70	70	2.453	0.029	1785712.4	198	0.011088012	0.99988912
300	80	80	2.805	0.017	2040814.4	151	0.007399007	0.99992601
300	90	90	3.157	0.005	2295916.4	136	0.005923561	0.999940764
300	100	100	3.51	0.03	2551018.4	120	0.004704004	0.99995296

TABLE III
NETWORK PERFORMANCE OF OVS WITH VARYING TIME

Time Allotted (Seconds)	Bandwidth Allotted (Mbps)	Bandwidth Utilized (Mbps)	Transfer Data (GB)	Jitter (ms)	Datagrams Sent(bytes)	Datagrams Lost(bytes)	Packet Loss (Percentage)	PDR (Packet Delivery Ratio)
300	100	95.7	3.34	0.0585	2550797	107679	4.221386492	0.957786
600	100	96.5	6.48	0.067	4908267	174957	3.564537137	0.964355
900	100	95.3	9.62	0.0755	7265737	242235	3.333935704	0.966661
1200	100	96.1	12.76	0.084	9623207	309513	3.216318635	0.967837
1500	100	95.9	15.9	0.0925	11980677	376791	3.144989219	0.96855

TABLE IV
NETWORK PERFORMANCE OF P4 ENABLED SWITCH WITH VARYING TIME

Time Allotted (Seconds)	Bandwidth Allotted (Mbps)	Bandwidth Utilized (Mbps)	Transfer Data (GB)	Jitter (ms)	Datagrams Sent(bytes)	Datagrams Lost(bytes)	Packet Loss (Percentage)	PDR (Packet Delivery Ratio)
300	100	100	3.49	0.082	2551030	123	0.004821582	0.999952
600	100	100	6.73	0.036	3762446	238	0.006325672	0.999937
900	100	100	9.97	0.107	4973862	353	0.007097101	0.999929
1200	100	100	13.21	0.095	6185278	468	0.007566354	0.999924
1500	100	100	16.45	0.113	7396694	583	0.0078819	0.999921

Based on the results provided in the tables, here are some insights:

Table I: Network Performance of OVS with Varying Bandwidth

- As the allotted bandwidth increases from 10 Mbps to 100 Mbps, the achieved bandwidth (Bandwidth Mbps) also increases accordingly.
- The amount of data transferred (Transfer Data GB) increases with higher bandwidth allocations.
- Jitter remains relatively low across different bandwidth allocations, indicating stable latency.

- The packet loss percentage (Packet Loss Percentage) increases slightly with higher bandwidth allocations.
- The packet delivery ratio (PDR) decreases slightly as the bandwidth allocation increases.

Table II: Network Performance of P4 Enabled Switch with Varying Bandwidth

- Like Table I, increasing the allocated bandwidth results in a higher achieved bandwidth.
- The amount of data transferred increases with higher bandwidth allocations.
- Jitter values remain relatively low across different bandwidth allocations.
- The packet loss percentage is extremely low for the P4 enabled switch, indicating excellent packet delivery.
- The packet delivery ratio remains consistently high across different bandwidth allocations.

Table III: Network Performance of OVS with Varying Time

- As the allotted time increases, the bandwidth utilized (Bandwidth Utilized Mbps) remains relatively stable.
- The amount of data transferred increases with longer time allocations.
- Jitter values remain relatively low for different time allotments.
- The packet loss percentage is relatively low across different time allotments.
- The packet delivery ratio remains consistently high for OVS.

Table IV: Network Performance of P4 Enabled Switch with Varying Time

- Like Table III, the bandwidth utilized remains consistently high for different time allotments.
- The amount of data transferred increases with longer time allocations.
- Jitter values remain relatively low for different time allotments.
- The packet loss percentage is extremely low for the P4 enabled switch, indicating excellent packet delivery.
- The packet delivery ratio remains consistently high across different time allotments.

Overall, the results indicate that both OVS and the P4 enabled switch to perform well in terms of network performance. The P4 enabled switch demonstrates lower packet loss and higher packet delivery ratios compared to OVS, showcasing its superior performance. Additionally, increasing the allocated bandwidth or time generally leads to higher achieved bandwidth and data transfer, while maintaining low jitter and packet loss.

C. Results: SFC Orchestration Time

The SFC orchestration time was measured for both the existing and proposed solutions, considering different use cases in the OpenStack cloud platform. The use cases represent various scenarios of SFC deployments, and the service functions were implemented accordingly. The use cases are as follows:

- Use Case 1: Multiple Ubuntu servers in the same network.
- Use Case 2: Ubuntu servers and Load balancer in the same network.
- Use Case 3: Ubuntu servers, Load balancer, and Firewall function in the same network.
- Use Case 4: Ubuntu servers, Load balancer, Firewall, and DNS function in the same network.
- Use Case 5: Two Ubuntu servers, one Load balancer, two private networks, two subnet allocations, and one router.
- Use Case 6 and Use Case 7: Increased number of users simultaneously requesting SFC request for Use Case 5.

Table V presents the results of the orchestration performance for the developed use cases. The overall orchestration time is calculated by noting the start time when the client sends the request and the stop time when the SFC request is successfully deployed, along with verifying that all the instances have reached the 'ACTIVE' status. The SFC orchestration time is determined by calculating the time difference between the stop time and the start time. Table V provides the gRPC and REST-based SFC orchestration time for both the OVS and P4 platforms.

TABLE V
SFC ORCHESTRATION TIME

	OVS		P4 Switch	
Use Case Number	REST-SFC (seconds)	gRPC-SFC (seconds)	REST-SFC (seconds)	gRPC-SFC (seconds)
1	9.88	8.33	8.4	6.88
2	15.65	11.82	13.49	8.66
3	21.32	14.31	18.58	10.44
4	28.43	17.8	23.67	12.22
5	27.43	13.49	23.61	12.62
6	58.67	28.22	50.87	26.84
7	120.81	58.71	104.26	54.06

Based on the results provided in Table V, we can derive the following insights:

1. Comparison between OVS and P4 Switch: Overall, the P4 Switch platform shows better SFC orchestration times compared to OVS. Across all use cases and communication protocols (REST-SFC and gRPC-SFC), the P4 Switch consistently demonstrates lower orchestration times. This suggests that the P4 Switch platform offers better performance and efficiency in SFC orchestration compared to OVS.

2. Impact of Use Case Complexity: The SFC orchestration time rises along with the use case complexity. The orchestration times for various use

cases within each platform and communication protocol can be compared to see this. The orchestration time for Use Case 1 is the shortest and Use Case 7 is the longest. This demonstrates that the quantity, complexity, and configuration of service functions affect the orchestration time.

3. Comparison between REST-SFC and gRPC-SFC: In comparison to REST-SFC, the gRPC-SFC communication protocol often shows faster orchestration times. Comparing the orchestration timings for the same use case and platform using various communication protocols will show this. According to the difference in orchestration times, gRPC-SFC appears to provide a more effective and quick communication method for SFC orchestration.

4. Scalability Considerations: Use Cases 6 and 7, where there are more simultaneous SFC requests, have much longer orchestration durations than the previous use cases. This emphasizes how crucial it is for SFC orchestration systems to take scalability and resource allocation into account. The orchestration time may be impacted when the number of requests and concurrent users rise, which could result in performance issues.

To develop and execute SFC orchestration systems with the best performance and efficiency, it is important to carefully consider the platform, communication protocol, and use case complexity. The results show SFC orchestration time for different combinations of platforms. The primary results show that proposed solution (gRPC based orchestration on P4 switch) is better than existing solution (REST based orchestration using OVS).

D. Network Performance Analysis: OVS and P4 Switch

The network performance is compared in terms of bandwidth utilized, transfer data, and packet delivery ratio (PDR). The performance analysis is generated using results obtained from Table I-IV.

Fig. 5 depicts the bandwidth utilization in both the OVS and P4 platforms when sending UDP packets for a duration of 300 seconds with varying allocated bandwidth. The analysis reveals that the P4 switch outperforms OVS in terms of bandwidth utilization. Particularly, as the allocated bandwidth size increases, the P4 switch demonstrates the highest utilization. Similarly, Fig. 6 illustrates the bandwidth utilization with a fixed allocation of 100 Mbps and varying time. The results demonstrate that the performance of the P4 switch surpasses that of OVS significantly in terms of bandwidth utilization.

Fig. 7 and Fig. 8 display the transfer data for varying bandwidth and time in both the OVS and P4 platforms. Initially, the bandwidth was kept constant at 300 Mbps, and by varying the time, it was observed that the P4 switch provided marginally higher transfer data compared to OVS. On the other hand, when the bandwidth allocation size was varied while keeping the time constant, it was evident that the P4 switch consistently outperformed OVS as the bandwidth increased.

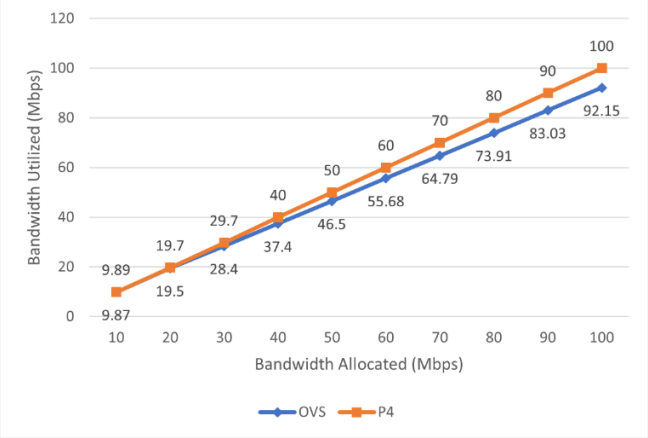

Fig. 5. Bandwidth Utilization with Varying Bandwidth Allotted

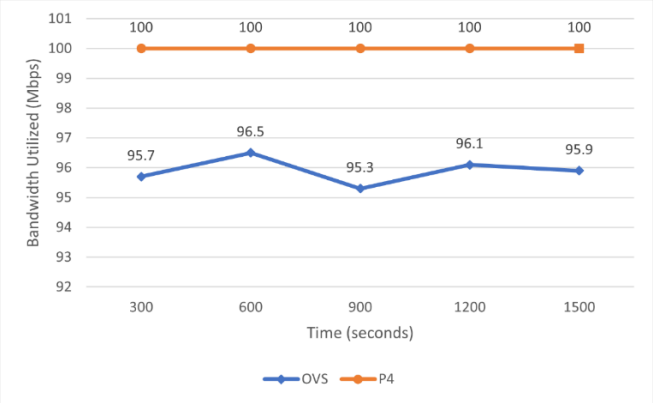

Fig. 6. Bandwidth Utilization with Varying Time

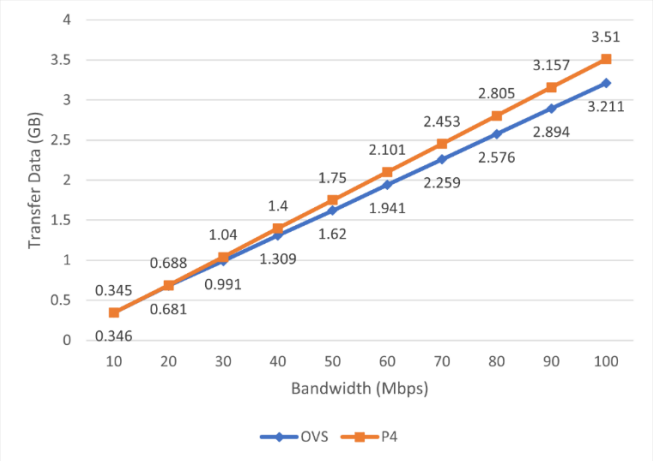

Fig 7. Transfer Data with Varying Bandwidth Allotted

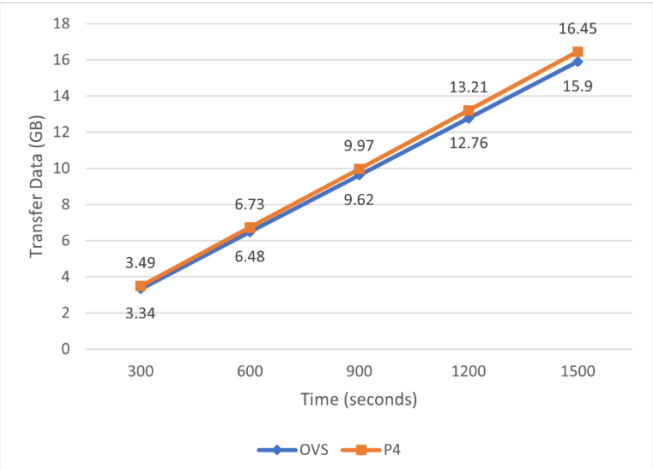

Fig. 8. Transfer Data with Varying Time

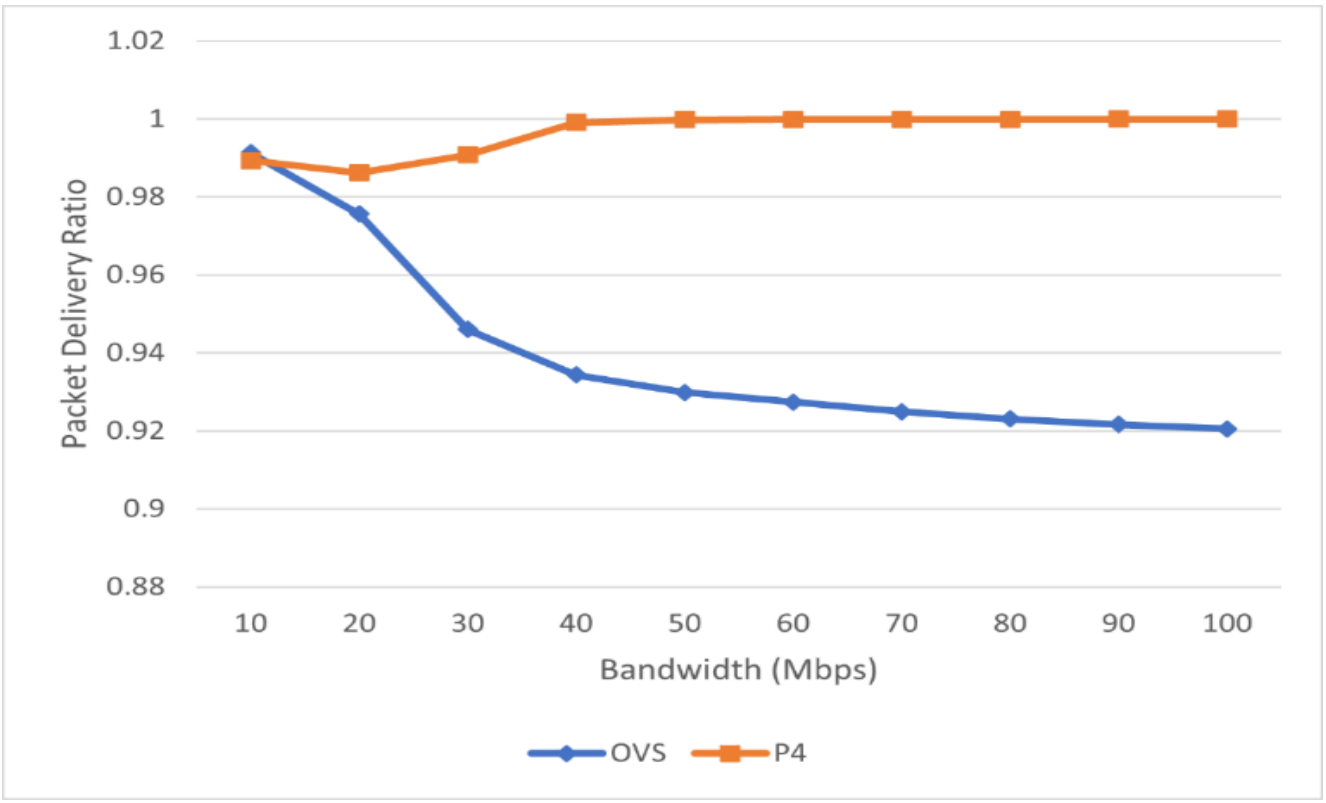

Fig. 9. Packet Delivery Ratio with Varying Bandwidth Allotted

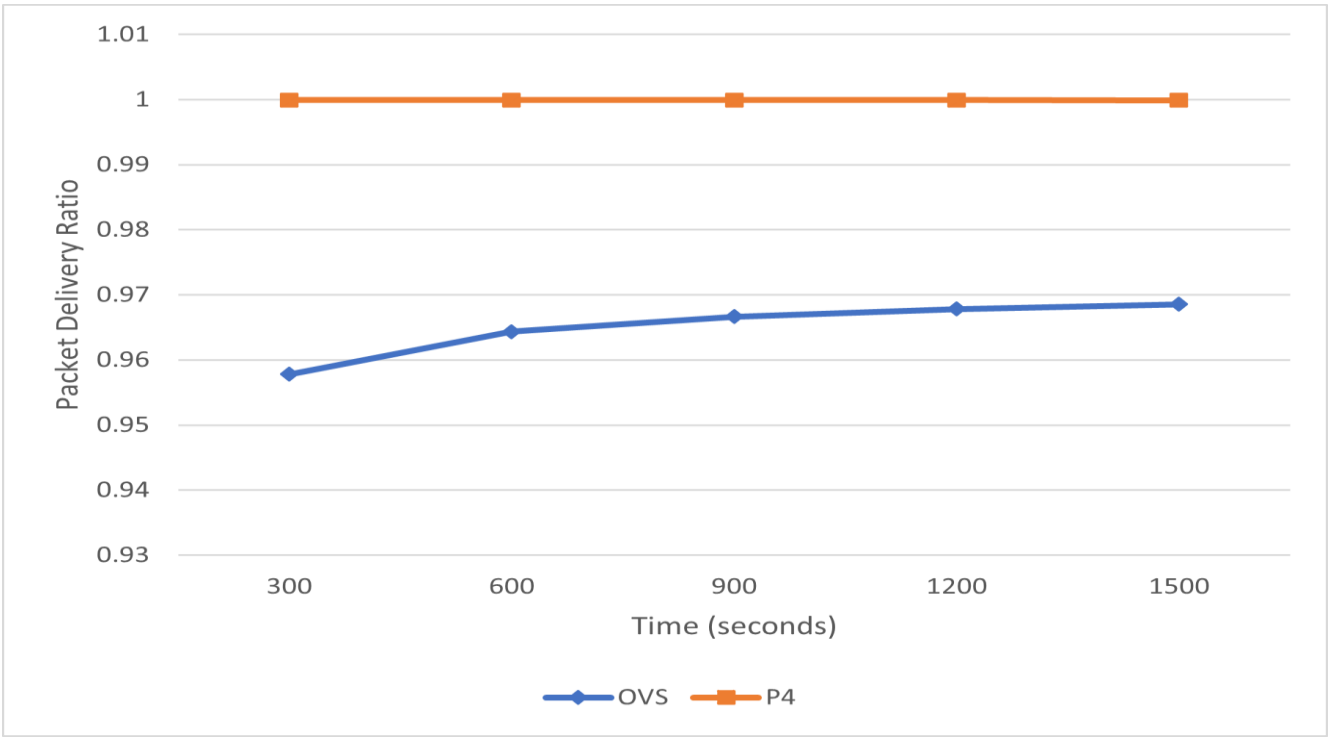

Fig. 10. Packet Delivery Ratio with Varying Time

PDR measures the percentage of successfully delivered packets to their intended destination. A higher PDR indicates better network performance and effective data transmission. Fig. 9 and Fig. 10 illustrate the PDR for sending UDP packets in the OVS and P4 platforms. Whether it is in varying time or varying allocated bandwidth, the P4 switch demonstrates a superior PDR compared to OVS, indicating improved network performance and reliable packet delivery.

E. SFC Orchestration Performance Analysis

SFC orchestration systems play a crucial role in rapidly creating and managing services in an efficient manner. The time required for creating a service in systems is known as

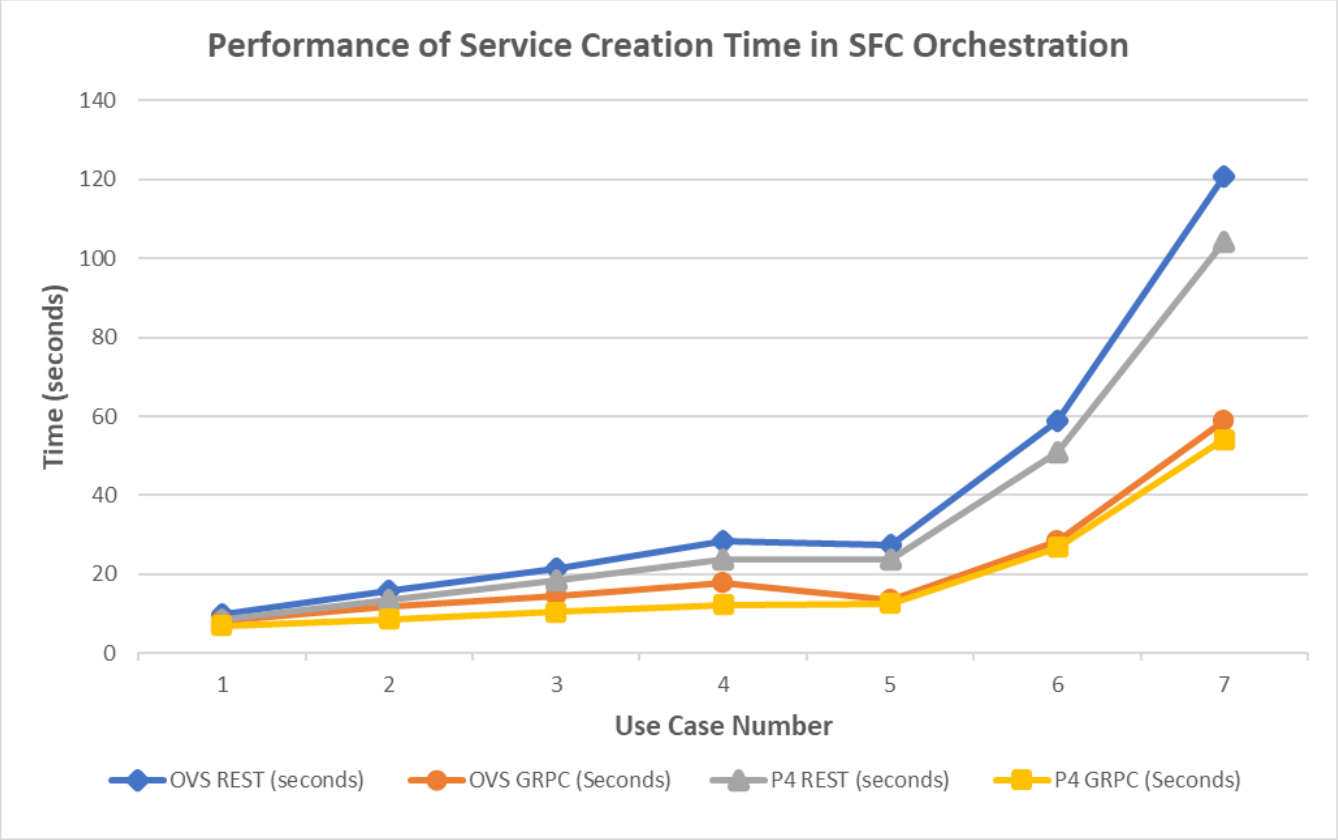

Fig. 11. Performance of Service Creation Time in SFC Orchestration for Developed Use Cases

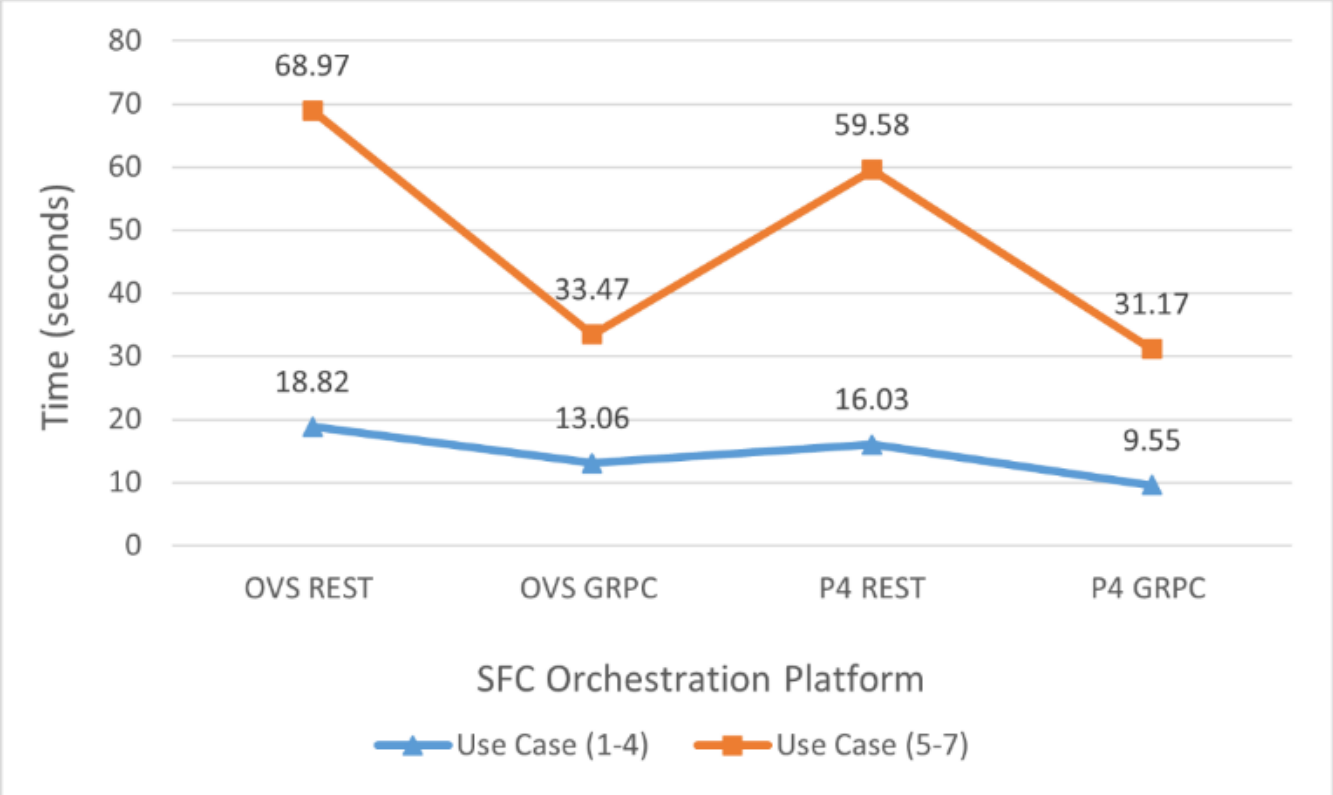

Fig. 12. Performance of Service Creation Time with SFC Orchestration Platforms

service creation time. In Fig. 11, the service creation time is depicted for each of the developed SFC use cases, utilizing four different platforms. These platforms are categorized as follows:

- Platform 1: REST-based SFC Orchestration using OVS (Existing Solution).
- Platform 2: gRPC-based SFC Orchestration using OVS.

- Platform 3: REST-based SFC Orchestration using P4 switch.
- Platform 4: gRPC-based SFC Orchestration using P4 switch.

The results presented in Fig. 11 indicate that the gRPC-based SFC orchestration (platform 4) exhibits superior service creation time compared to platforms 1, 2, and 3 (as mentioned in Table V). To gain a better understanding of the extent of improvement offered by the proposed solution, the percentage improvement in service creation time is calculated using the provided equation.

$$\% \ Improvement = ((OT-NT)/OT) * 100 \quad (1)$$

By using equation (1) and the results from Table V, the percentage of improvement in service creation time can be calculated by comparing the old time (OT) and new time (NT). The calculations yield the following results:

- In the OVS platform, gRPC-based SFC orchestration is 45.89% faster than REST-based SFC orchestration.
- In the P4 platform, gRPC-based SFC orchestration is 45.76% faster than REST-based SFC orchestration.
- REST-based SFC orchestration in the P4 platform is 13.93% faster than in the OVS platform.
- gRPC-based SFC orchestration in the P4 platform is 13.72% faster than in the OVS platform.

These percentages demonstrate the improvement in service creation time achieved by the different orchestration methods and platforms.

The results for the developed use cases are categorized based on the number of services communicated in the SFC request. They are divided into two groups: use case 1 to 4, which represent SFC requests excluding network services, and use case 5 to 7, which represent SFC requests including network services. The classification of the results is shown in Table VI.

TABLE VI
ORCHESTRATION TIME FOR SFC USE CASES

	OVS PLATFORM		P4 Switch Platform	
Use Case Number	OVS-REST (seconds)	OVS-gRPC (seconds)	P4-REST (seconds)	P4- gRPC (seconds)
1 to 7	40.31	21.81	34.69	18.81
1 to 4	18.82	13.06	16.03	9.55
5 to 7	68.97	33.47	59.58	31.17

In summary, Table VI showcases the orchestration time for different SFC use cases using both the OVS platform and the P4 Switch platform.

1. Use Cases 1 to 7:
   - In the OVS platform, the REST-based SFC orchestration takes 40.31 seconds, while the gRPC-based orchestration is faster at 21.81 seconds.
   - In the P4 Switch platform, the REST-based SFC orchestration takes 34.69 seconds, while the gRPC-based orchestration is even faster at 18.81 seconds.
2. Use Cases 1 to 4:
   - When considering only the first four use cases, the OVS platform with REST-based orchestration takes 18.82 seconds, whereas the gRPC-based orchestration reduces the time to 13.06 seconds.
   - Similarly, in the P4 Switch platform, the REST-based orchestration takes 16.03 seconds, while the gRPC-based orchestration achieves a faster time of 9.55 seconds.
3. Use Cases 5 to 7:
   - Focusing on the last three use cases, the OVS platform with REST-based orchestration requires 68.97 seconds, while the gRPC-based orchestration decreases the time to 33.47 seconds.
   - In the P4 Switch platform, the REST-based orchestration takes 59.58 seconds, and the gRPC-based orchestration further improves the time to 31.17 seconds.

These results highlight that the gRPC-based SFC orchestration performs better than the REST-based orchestration for both OVS and P4 Switch platforms. The P4 Switch platform shows improved orchestration times compared to the OVS platform, regardless of the orchestration method used.

Based on the provided information, the average orchestration time for grouped SFC use cases is shown in Fig. 12. Using equation (1), the average percentage improvement in service creation time can be calculated. The results are as follows:

- Excluding network services (Use Case 1-4):
  - In the OVS platform, gRPC-based SFC orchestration is 30.60% faster than REST-based SFC orchestration.
  - In the P4 platform, gRPC-based SFC orchestration is 40.42% faster than REST-based SFC orchestration.
  - REST-based SFC orchestration in the P4 platform is 14.82% faster than the OVS platform.

- gRPC-based SFC orchestration in the P4 platform is 26.87% faster than the OVS platform.

- Including network services (Use Case 5-7):
  - In the OVS platform, gRPC-based SFC orchestration is 51.47% faster than REST-based SFC orchestration.
  - In the P4 platform, gRPC-based SFC orchestration is 47.68% faster than REST-based SFC orchestration.
  - REST-based SFC orchestration in the P4 platform is 13.61% faster than the OVS platform.
  - gRPC-based SFC orchestration in the P4 platform is 6.87% faster than the OVS platform.

The analysis concludes by highlighting the main findings of the paper:

- Using the P4 switch, the network performance of the data plane in the OpenStack network has improved compared to OVS in terms of higher bandwidth utilization, transfer data, and packet delivery ratio.
- By utilizing P4 and gRPC API, the service creation time is approximately 50% faster compared to traditional orchestration using OVS and REST API on the OpenStack platform.
- For SFC use cases:
  - With P4 and gRPC API, SFC orchestration including network components is approximately 54.80% faster than traditional orchestration using OVS and REST API on the OpenStack platform.
  - Similarly, using P4 and gRPC API in SFC orchestration excluding network components is approximately 49.25% faster than traditional orchestration using OVS and REST API on the OpenStack platform.

Considering all SFC use cases, the overall performance indicates that gRPC-based SFC orchestration using the P4 switch is approximately 53.32% faster than REST-based SFC orchestration using OVS.

V. Conclusion and Future Work

This research paper introduced and evaluated the utilization of gRPC API and P4 switch in SFC orchestration within OpenStack. The findings demonstrate significant enhancements in network performance and service creation time. Specifically, the utilization of P4 switch resulted in a notable 7% improvement in bandwidth utilization and a corresponding 6% enhancement in packet delivery ratio compared to OVS. Furthermore, the orchestration services on the P4 platform exhibited a 13% improvement in creation time for scenarios involving network services. The combination of gRPC API and P4 switch outperformed the REST API and OVS in terms of service creation time, showcasing a remarkable 50% improvement. The study concludes that both gRPC API and P4 switch offer superior options for SFC orchestration in OpenStack. The gRPC API enables faster and more efficient communication, while the P4 switch enhances packet delivery. Incorporating the P4 switch with gRPC API into the orchestration strategy can enhance communication in OpenStack, with a specific focus on improving energy efficiency. The measures employed in the SFC orchestration design using gRPC API with P4 can also be adopted in other cloud orchestration systems to enhance energy efficiency.

References

[1] Carretero, J. and Blas, J.G., 2014. Introduction to cloud computing: platforms and solutions. Cluster computing, 17(4), pp.1225-1229.

[2] Cox, J.H., Chung, J., Donovan, S., Ivey, J., Clark, R.J., Riley, G. and Owen, H.L., 2017. Advancing software-defined networks: A survey. IEEE Access, 5, pp.25487-25526.

[3] Mijumbi, R., Serrat, J., Gorricho, J.L., Bouten, N., De Turck, F. and Boutaba, R., 2015. Network function virtualization: State-of-the-art and research challenges. IEEE Communications surveys & tutorials, 18(1), pp.236-262.

[4] Sefraoui, O., Aissaoui, M. and Eleuldj, M., 2012. OpenStack: toward an open-source solution for cloud computing. International Journal of Computer Applications, 55(3), pp.38-42.

[5] Adoga, H.U. and Pezaros, D.P., 2022. Network Function Virtualization and Service Function Chaining Frameworks: A Comprehensive Review of Requirements, Objectives, Implementations, and Open Research Challenges. Future Internet, 14(2), p.59.

[6] Pfaff, B., Pettit, J., Koponen, T., Jackson, E., Zhou, A., Rajahalme, J., Gross, J., Wang, A., Stringer, J., Shelar, P. and Amidon, K., 2015. The Design and Implementation of Open {vSwitch}. In 12th USENIX symposium on networked systems design and implementation (NSDI 15) (pp. 117-130)

[7] Petrillo, F., Merle, P., Moha, N. and Guéhéneuc, Y.G., 2016, October. Are REST APIs for cloud computing well-designed? An exploratory study. In International Conference on Service-Oriented Computing (pp. 157-170). Springer, Cham.

[8] Kaur, K., Mangat, V. and Kumar, K., 2022. A review on Virtualized Infrastructure Managers with management and orchestration features in NFV architecture. Computer Networks, p.109281.

[9] Bosshart, P., Daly, D., Gibb, G., Izzard, M., McKeown, N., Rexford, J., Schlesinger, C., Talayco, D., Vahdat, A., Varghese, G. and Walker, D., 2014. P4: Programming protocol-independent packet processors. ACM SIGCOMM Computer Communication Review, 44(3), pp.87-95.

[10] da Costa Cordeiro, W.L., Marques, J.A. and Gaspary, L.P., 2017. Data plane programmability beyond openflow: Opportunities and challenges for network and service operations and management. Journal of Network and Systems Management, 25(4), pp.784-818.

[11] Liatifis, A., Sarigiannidis, P., Argyriou, V. and Lagkas, T., 2022. Advancing SDN: from OpenFlow to P4, a Survey. ACM Computing Surveys (CSUR).

[12] Open vSwitch. (n.d.). Open vSwitch. [online] Available at: https://www.openvswitch.org/ [Accessed 15 Jan 2023].

[13] P4 Language Consortium. (n.d.). P4 Language. [online] Available at: https://p4.org/ [Accessed 15 Jan 2023].

[14] OpenStack Foundation. (n.d.). OpenStack: The open-source cloud computing platform. [online] Available at: https://www.openstack.org/ [Accessed 15 Jan 2023].

[15] P4.org. (n.d.). P4 Switch. [online] Available at: https://p4.org/p4-switches/ [Accessed 15 Jan 2023].

[16] grpc.io. (n.d.). gRPC: A high-performance, open-source framework for building remote procedure calls. [online] Available at: https://grpc.io/ [Accessed 15 Jan 2023].

[17] Callegati, F., Cerroni, W., Contoli, C. and Santandrea, G., 2014, October. Performance of Network Virtualization in cloud computing infrastructures: The OpenStack case. In 2014 IEEE 3rd International Conference on Cloud Networking (CloudNet) (pp. 132-137). IEEE.
[18] Callegati, F., Cerroni, W. and Contoli, C., 2016. Virtual networking performance in openstack platform for network function virtualization. Journal of Electrical and Computer Engineering, 2016.
[19] Foresta, F., Cerroni, W., Foschini, L., Davoli, G., Contoli, C., Corradi, A. and Callegati, F., 2018, May. Improving OpenStack networking: Advantages and performance of native SDN integration. In 2018 IEEE International Conference on Communications (ICC) (pp. 1-6). IEEE.
[20] Huang, Y.X. and Chou, J., 2020. Evaluations of Network Performance Enhancement on Cloud-native Network Function. In Proceedings of the 2021 on Systems and Network Telemetry and Analytics (pp. 3-8).
[21] Chung, W.C. and Wang, Y.H., 2022. The Effects of High-Performance Cloud System for Network Function Virtualization. Applied Sciences, 12(20), p.10315.
[22] Franco, D., Atutxa, A., Sasiain, J., Ollora, E., Higuero, M., Astorga, J. and Jacob, E., 2022. Towards integrating hardware Data Plane acceleration in Network Functions Virtualization. arXiv preprint arXiv:2203.10920.
[23] Mandal, P. and Jain, R., 2019, December. Comparison of openstack orchestration procedures. In 2019 International Conference on Smart Applications, Communications and Networking (SmartNets) (pp. 1-4). IEEE.
[24] Petrillo, F., Merle, P., Moha, N. and Guéhéneuc, Y.G., 2016, October. Are REST APIs for cloud computing well-designed? An exploratory study. In International Conference on Service-Oriented Computing (pp. 157-170). Springer, Cham.
[25] Adoga, H.U. and Pezaros, D.P., 2022. Network Function Virtualization and Service Function Chaining Frameworks: A Comprehensive Review of Requirements, Objectives, Implementations, and Open Research Challenges. Future Internet, 14(2), p.59.