



RESEARCH ARTICLE

OPEN ACCESS

DESIGN AND IMPLEMENTATION OF A PROPOSED SDN-VXLAN ARCHITECTURE FOR ENTERPRISE NETWORK

Daniel Nuñez-Agurto^{*1}, Luis Paredes-Alcivar², John Cruz-Garzon³ and Franklin Perez⁴

^{1, 2, 3, 4}Department of Computer Science, Universidad de las Fuerzas Armadas ESPE, Sangolquí, Ecuador.

¹<https://orcid.org/0000-0001-7737-3815>, ²<https://orcid.org/0000-0002-5568-509X>,

³<http://orcid.org/0009-0006-7550-9962>, ⁴<https://orcid.org/0009-0009-4974-571X>

Email: adnunez1@espe.edu.ec

ARTICLE INFO

Article History

Received: February 27, 2026

Reviewed: June 11, 2026

Accepted: July 11, 2026

Published: July 31, 2026

Keywords:

Software-Defined Networking (SDN),

Virtual Extensible LAN (VxLAN),

PPDIOO,

Network Architecture,

Open vSwitch (OVS).

ABSTRACT

Efficient and scalable network architectures are essential for enterprise environments that demand reliable and high-performance connectivity. Virtual Extensible LAN (VXLAN) technology enables the encapsulation of Layer 2 traffic over Layer 3 infrastructures, while Software-Defined Networking (SDN) introduces centralized control and programmability by decoupling the control and data planes. This study presents the design and implementation of an enterprise network architecture that integrates SDN and VXLAN technologies to enhance scalability, performance, and security. The PPDIIO lifecycle methodology was adopted as a structured framework for design, deployment, and optimization. The proposed architecture was implemented and validated in a GNS3 emulated environment using OpenDaylight as the SDN controller and Open vSwitch (OVS) as the virtual switching layer. Functional testing, including connectivity verification and packet flow analysis, confirmed correct operation of VXLAN tunneling under SDN control, demonstrating the feasibility and effectiveness of the proposed model. The results highlight that the integration of SDN and VXLAN provides a flexible, programmable, and secure foundation for next-generation enterprise network infrastructures.



Copyright ©2026 by authors and Galileo Institute of Technology and Education of the Amazon (ITEGAM). This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).

I. INTRODUCTION

In modern business environments, connectivity and performance are essential to ensuring operational efficiency and competitiveness. In an increasingly digitalized global environment marked by the rapid advancement of smart devices, the volume of data traversing networks is growing exponentially, leading to greater complexity in network infrastructures [1], [2]. A production network typically comprises numerous devices, operates under various protocols, and supports a wide range of applications [3]. The adoption of emerging technologies such as Virtual Extensible LAN (VXLAN) [4], [5] and Software-Defined Networking (SDN) [6], [7] represents an innovative alternative that significantly improves performance compared to traditional networks. Thus, Internet Service Providers (ISPs) and network administrators must adjust to these transformations by implementing the right tools and methodologies to guaranty effective network management. SDN is a dynamic and evolving architecture that decouples the control plane from the data plane, making it particularly suitable for the changing requirements of modern applications [8], [9]. A centralized controller offers a global view of the network, enabling enhanced monitoring, configuration, and packet flow analysis. Meanwhile, VXLAN enables the creation of a large number of isolated virtual networks, overcoming limitations of traditional VLAN technologies [10], [11]. This tunneling protocol facilitates network segmentation around virtual machines and allows the deployment of Layer 2 networks over a Layer 3 infrastructure, thereby improving overall traffic performance. The integration of SDN and VXLAN brings together essential capabilities required to overcome the limitations of conventional networking technologies. By [12] proposed an SDN-controlled VXLAN architecture implemented in an intelligent data center to support seamless virtual machine migration. They deployed the design using Mininet and achieved optimal load balancing results. Their findings demonstrated that an SDN-based VXLAN architecture reduces IP multicast resource usage and meets deployment requirements in cloud computing environments. Similarly [13] evaluated SDN-controlled VXLAN in multi-site enterprise networks using Mininet, identifying high traffic volumes and latency as primary challenges.

They proposed dynamic VXLAN tunnels, tunnel division for hybrid and corporate access, and centralized policy management to improve performance and operational agility. These results highlight the potential of SDN and VXLAN to optimize complex infrastructures and pave the way for AI-driven solutions in network security. Network performance and efficiency are critical for service interconnection in enterprise environments. However, few works have addressed the systematic design and functional deployment of SDN-VXLAN architectures in enterprise contexts, where interoperability with existing infrastructures and operational frameworks is critical. To overcome these limitations, this study aims to design an efficient network architecture that integrates SDN and VXLAN to optimize enterprise network performance. This study aims to design and implement a functional enterprise network architecture that integrates SDN and VXLAN technologies, following the structured PPDIIO (Prepare, Plan, Design, Implement, Operate, Optimize) lifecycle methodology proposed by Cisco. The primary contribution of this work lies in demonstrating a replicable and verifiable implementation framework that validates the interoperability and feasibility of SDN-VXLAN integration over an emulated MPLS underlay network. The proposed model was implemented using the GNS3 network emulator, with OpenDaylight serving as the SDN controller and Open vSwitch (OVS) as the virtual switching element within Ubuntu-based Docker hosts. Functional testing focused on verifying VXLAN tunneling, OpenFlow-based flow control, and SDN management capabilities. The remainder of this paper is organized as follows: Section 2 describes the materials and methods applied throughout the PPDIIO lifecycle. Section 3 presents the main implementation results and discusses the observed operational behavior. Finally, Section 4 summarizes the key findings and outlines directions for future research.

II. RELATED WORKS

The integration of SDN with network virtualization mechanisms such as VXLAN and EVPN has led to a substantial body of research aimed at enhancing automation, scalability, and traffic isolation across diverse scenarios, ranging from multi-tenant data centers and multisite infrastructures to educational networks and domain-specific applications. However, most of these studies concentrate on isolated data-plane optimizations, energy-efficiency improvements, or narrowly scoped use cases, and therefore do not provide a comprehensive methodology or a reproducible design suitable for enterprise environments. To contextualize the contribution of our work, we present a detailed comparison of ten relevant studies that explore different combinations of SDN controllers, VXLAN/EVPN technologies, underlay configurations, validation environments, and design objectives, as summarized in Table 1.

Table 1: Comparative summary of related works and the proposed SDN–VXLAN architecture.

Ref.	Objective	Technologies	Validation	Contributions	Limitations
[14]	Performance optimization	SDN + P4 switches + EVPN/VXLAN	Experimental tests	Accelerated user-plane performance; ARP suppression; multi-tenancy and DAG support	Lacks implementation methodology; limited to data-plane optimization
[15]	Eco-efficient multi-tenant datacenter management	Ryu controller + VXLAN	Experimental	Zero traffic leakage; automated VTEP discovery; energy-efficient orchestration	No detailed deployment methodology; limited evaluation scenarios
[16]	Green multisite datacenter orchestration	SDN + EVPN/VXLAN + DPDK	Simulation	Reduced workload migration time (4.24%); resilient multi-homing architecture	Specific to GMDC; not suitable for enterprise networks
[17]	Distance learning enhancement	SDN + VXLAN	Practical scenarios	Remote access as local network; improved STEM resource availability	Not generalizable to enterprise environments; lacks benchmarking
[18]	Smart agriculture / online learning	SDN + VXLAN	Conceptual study	IoT connectivity through overlays; supports future national datacenters	No large-scale validation; conceptual-only
[19]	IPv6 deployment in campus network	SDN campus controller + VXLAN	Conceptual design	Automated IPv6 assignment; unified policy distribution	No empirical validation; lacks performance evaluation
[20]	Cloud-network collaboration	SDN orchestrator + VXLAN	Analytical review	Unified multi-region cloud-network management framework	Highly abstract; no implementation/testing
[21]	AD-Campus optimization	SeerEngine-Campus + EVPN/VXLAN	Simulation	Step-by-step deployment; service separation across campuses	Vendor-specific; no technical comparison
[22]	Traffic load balancing in overlays	SDN + VXLAN tunneling	Experimental	Reduced latency and access time; efficient link usage	Limited to load balancing; no full architectural design
[23]	Hybrid distance learning	Floodlight + VXLAN	Real GNS3 tests	Redundancy, VPN integration; proven in real scenarios	Education-focused; not enterprise-oriented
Our Proposed	Enterprise network design	Ryu + OVS + VXLAN overlay	GNS3/Mininet emulation	PPDIIO-based methodology; reproducible enterprise architecture	Does not target performance benchmarking; design-focused

Source: Authors, (2026).

By [14] present an SDN-enabled EVPN-VXLAN architecture targeting geo-distributed sites and accelerate the user plane using P4-programmable switches. Their design integrates SDN host detection and EVPN host learning for ARP suppression, supports multi-tenancy and a distributed anycast gateway model, and demonstrates reduced host communication latency and improved user-plane efficiency.

This work shows how programmable data-plane elements can significantly enhance EVPN-VXLAN deployments across dispersed sites, complementing but differing from the present study which focuses on a fully documented SDN-VXLAN implementation methodology for enterprise networks. In turn [15] propose an eco-efficient SDN-VXLAN architecture for multi-tenant datacenters that leverages an optimized Ryu controller to automate VTEP discovery and enforce perfect tenant isolation. Their experiments report automated VTEP configuration, 0% cross-tenant leakage, and optimized resource utilization-framed as a sustainable, centralized approach for cloud providers. While directly comparable in its use of SDN+VXLAN, the focus on eco-efficiency and controller optimizations complements our work's emphasis on a reproducible PPDIIOO-based SDN-VXLAN implementation for enterprise networks. By [16] introduce the Green Multisite Data Center (GMDC) architecture that leverages SDN with EVPN/VXLAN and DPDK-accelerated data-plane techniques to support renewable-powered, geo-distributed datacenters. Their simulations show a 4.24% reduction in total workload migration time and improved resiliency via all-active paths and multi-homing. The GMDC work provides an eco-centric perspective on multisite overlay networks, offering complementary design goals to our enterprise-focused SDN-VXLAN implementation. According to [17] apply VXLAN and SDN to optimize Mali's national fiber infrastructure for distance learning, enabling students to access resources as if on a local network. Their implementation demonstrates improved accessibility and remote collaboration for STEM education. Although targeted to educational infrastructure, the study exemplifies practical VXLAN overlay deployment over operator networks—insights useful when comparing deployment choices in enterprise SDN-VXLAN proposals. By [18] investigate the use of VXLAN and SDN to support online learning and IoT-enabled smart agriculture in the Central African Republic. Their case study outlines how overlays can enable connectivity and data management when a national datacenter is absent. The work illustrates the adaptability of SDN-VXLAN solutions to resource-constrained environments, complementing enterprise architectures by highlighting deployment considerations in low-infrastructure contexts.

In turn [19] proposes an SDN-based IPv6 deployment scheme using VXLAN to decouple control and user planes in a campus network, enabling attribute-based IPv6 address allocation and policy enforcement for location-agnostic access. The scheme is directly relevant for enterprise/campus scenarios that require seamless IPv6 transition and overlay segmentation, offering architectural mechanisms that complement our PPDIIOO-guided SDN-VXLAN implementation. According to [20] examine the deployment of white-box network equipment combined with SDN and VXLAN in large-scale cloud environments, proposing a collaborative operation platform for unified management across multi-region clouds. Their work is valuable for understanding operational and orchestration implications when adopting SDN-VXLAN at scale, informing enterprise implementation choices in our study. By [21] detail an AD-Campus solution employing EVPN-VXLAN and a SeerEngine SDN controller to enable scalable, multi-transport campus networks. Their simulation cycle and deployment procedures offer practical guidance on EVPN-VXLAN rollouts and demonstrate service separation benefits—useful operational context for our PPDIIOO-based SDN-VXLAN implementation. In turn [22] implement an SDN-driven traffic balancer over VXLAN overlay networks, demonstrating decreased access times and improved link utilization. Their contribution provides a concrete mechanism for overlay traffic engineering that can be adopted within enterprise SDN-VXLAN designs to enhance availability and performance. By [23] present a hybrid distance-learning infrastructure that overlays VXLAN over national DWDM networks and manages flows with a Floodlight SDN controller. Their GNS3 experiments and VPN integrations demonstrate practical deployment steps for VXLAN overlays in real operational environments, informing controller selection and deployment workflows relevant to enterprise VXLAN implementations. Based on the above, these studies collectively demonstrate the versatility of SDN-VXLAN and EVPN-based architectures across a wide range of domains, including multisite datacenters, educational infrastructures, cloud environments, and resource-constrained deployments. However, despite their individual contributions, none of them provides a comprehensive, replicable, and controller-agnostic methodology specifically tailored to enterprise networks. This gap motivates the present work, which delivers a complete PPDIIOO-based SDN-VXLAN design and implementation blueprint, supported by controlled emulation and practical deployment guidelines.

III. MATERIALS AND METHODS

This study is structured around the PPDIIOO methodology, chosen for its systematic and comprehensive approach to network design and implementation [24]. Its adoption is motivated by the need to evaluate the impact of VXLAN technology on enterprise network performance in comparison to traditional architectures. Originating from Cisco's network lifecycle framework, the PPDIIOO model is divided into six sequential phases: Prepare, Plan, Design, Implement, Operate, and Optimize. The following subsections detail the development of this study according to each phase of the methodology, demonstrating its relevance and effectiveness in guiding the deployment of an SDN-based VXLAN architecture.

III.1 PREPARE

This phase focuses on identifying and selecting the technological components required to meet the performance, scalability, and security requirements of the proposed enterprise network architecture. The analysis considered key functional requirements, including support for Layer 2 traffic over a Layer 3 infrastructure, separation of control and data planes through SDN principles, and the integration of VXLAN to enhance segmentation and scalability within an overlay framework. For network emulation and validation, GNS3 was selected as the simulation platform, available as a desktop application and as an Ubuntu-based virtual machine server [25]. The server environment was configured with 5 GB of RAM to ensure stable operation under multiple virtualized components. Cisco IOU images were adopted as network devices due to their modularity, scalability, and suitability for WAN/MAN service aggregation scenarios [26]. Open vSwitch (OVS) was implemented as the virtual switching fabric, providing OpenFlow compatibility and support for advanced monitoring protocols such as NetFlow, sFlow, 802.1ag, and IPFIX [27], [28]. Supporting components included Ubuntu-based Docker containers and a lightweight QEMU Firefox image obtained from the official GNS3 repository [29] for host emulation and traffic validation. The SDN control plane was managed by OpenDaylight (Karaf distribution 0.5.2 Boron SR2), selected for its modular architecture and compatibility with complex network environments [30].

This technological preparation establishes a controlled and reproducible environment for the subsequent planning, design, and implementation phases, enabling structured integration of SDN and VXLAN mechanisms within the proposed enterprise architecture.

III.2 PLAN

Based on the technologies outlined during the preparation phase, a structured design plan was developed for the SDN-based network architecture. Since this project involves a network emulation, the intended users are virtual hosts. For this implementation, a VXLAN tunnel was established between two endpoint nodes. However, it is important to note that VXLAN technology supports the creation of up to 16,000,000 Virtual Network Identifiers (VNIs), each providing an independent segmentation space, thus offering a high degree of scalability for future deployments. The plan includes the specific tasks required to complete the project, as detailed in Table 2. These activities were distributed across a four-month timeline: during the first month, the preparation and planning phases were conducted; the second month was dedicated to network design and implementation; the third month focused on the operation phase; and finally, the fourth month involved optimization of the emulated network, incorporating necessary adjustments based on functional validation.

Table 2: Work plan for the implementation of SDN with VXLAN.

Phase	Month 1	Month 2	Month 3	Month 4
Preparation	X			
Planning	X			
Network Design		X		
Implementation		X		
Operation			X	
Optimization				X

Source: Authors, (2026).

III.3 PLAN

Based on the information gathered in the previous stages, the network topology was designed. It consists of an MPLS underlay network composed of two P routers and two PE routers, along with two CE routers located at different sites. Each of these includes an Open vSwitch container, an OpenDaylight controller, a switch, and a Ubuntu Docker Guest host. The VXLAN overlay network is configured between the virtual Open vSwitch instances, as shown in Figure 1.

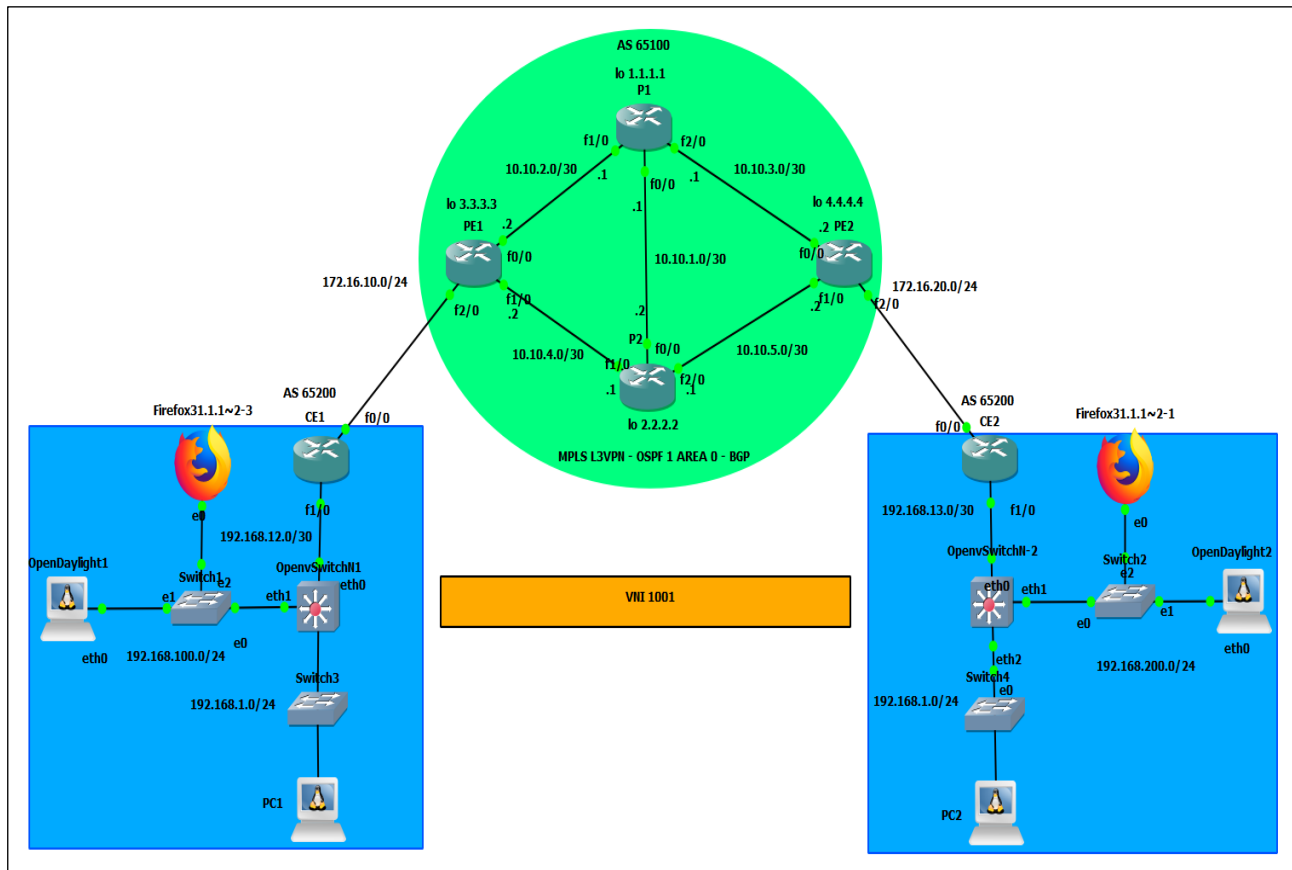


Figure 1: Test network topology.

Source: Authors, (2026).

Based on the established topology, IP addressing is assigned to each device, as detailed in Table 3.

Table 3: IP addressing of network devices.

Device	Interface	IP Address	Subnet Mask	Gateway
P1	Lo0	1.1.1.1	255.255.255.255	—
	Fa0/0	10.10.1.1	255.255.255.252	—
	Fa1/0	10.10.2.1	255.255.255.252	—
	Fa2/0	10.10.3.1	255.255.255.252	—
P2	Lo0	2.2.2.2	255.255.255.255	—
	Fa0/0	10.10.1.2	255.255.255.252	—
	Fa1/0	10.10.4.1	255.255.255.252	—
	Fa2/0	10.10.5.1	255.255.255.252	—
PE1	Lo0	3.3.3.3	255.255.255.255	—
	Fa0/0	10.10.2.2	255.255.255.252	—
	Fa1/0	10.10.4.2	255.255.255.252	—
	Fa2/0	172.16.10.1	255.255.255.0	—
PE2	Lo0	4.4.4.4	255.255.255.255	—
	Fa0/0	10.10.3.2	255.255.255.252	—
	Fa1/0	10.10.5.2	255.255.255.252	—
	Fa2/0	172.16.20.1	255.255.255.0	—
CE1	Lo0	172.16.1.10	255.255.255.255	—
	Fa0/0	172.16.10.2	255.255.255.0	—
	Fa1/0	192.168.12.1	255.255.255.252	—
CE2	Lo0	172.16.2.20	255.255.255.255	—
	Fa0/0	172.16.20.2	255.255.255.0	—
	Fa1/0	192.168.13.1	255.255.255.252	—
Open vSwitch 1	Eth0	192.168.12.2	255.255.255.252	192.168.12.1
	Eth1	192.168.100.1	255.255.255.0	—
	Eth2	192.168.1.1	255.255.255.0	—
Open vSwitch 2	Eth0	192.168.13.2	255.255.255.252	192.168.13.1
	Eth1	192.168.200.1	255.255.255.0	—
	Eth2	192.168.1.1	255.255.255.0	—
OpenDaylight 1	Eth0	192.168.100.2	255.255.255.0	192.168.100.1
OpenDaylight 2	Eth0	192.168.200.2	255.255.255.0	192.168.200.1
PC1 (Docker Guest)	E0	192.168.1.2	255.255.255.0	192.168.1.1
PC2 (Docker Guest)	E0	192.168.1.3	255.255.255.0	192.168.1.1

Source: Authors, (2026).

III.4 IMPLEMENT

III.4.1 WAN Configuration

The WAN network implementation began with the basic configuration of IP connectivity among the core nodes of the Underlay network. Subsequently, dynamic routing was established using the OSPF protocol to ensure network convergence and device reachability. Following this, MPLS functionality was enabled to allow label switching on the core routers. The Label Distribution Protocol (LDP) was configured to automate label exchange between devices, and specific label ranges were defined for label management. Once the basic configuration of the core routers (P routers) was completed, the implementation of the Provider Edge (PE) routers was carried out. On these devices, IP addressing was assigned according to the established design, and Virtual Routing and Forwarding (VRF) instances were configured to provide logical network segmentation, as shown in Figure 2.. Each VRF was associated with a unique identifier (Route Distinguisher) and import/export route policies (Route Targets), allowing client isolation and facilitating multipoint routing.

```

PE1#sh run vrf
Building configuration...

Current configuration : 352 bytes
ip vrf CLIENTEA
 rd 65200:1
 route-target export 65200:1
 route-target import 65200:2
!
```

Figure 2: Figure title.

Source: Authors, (2026).

OSPF was also enabled on the PE routers to advertise internal networks and support MPLS label exchange. The outcome of this configuration is shown in Figure 3. Additionally, the BGP protocol was configured to establish sessions between edge nodes and enable the transport of VPNv4 routes, which are necessary for inter-site communication over the MPLS backbone.

Extended capabilities were activated to ensure the proper transport of community attributes among BGP neighbors. Within the context of the implemented VRF, specific BGP sessions were also configured to establish connectivity with Customer Edge (CE) routers, thus guaranteeing correct end-to-end route distribution. Finally, the CE routers were configured with BGP routing capabilities, enabling direct interaction with their respective PE routers. Internal routes were advertised, and neighbor relationships were established with the edge routers. The CE router configuration was replicated at both ends of the network to maintain symmetry in the topology.

```

PE1#sh ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
+ - replicated route, % - next hop override

Gateway of last resort is not set

1.0.0.0/32 is subnetted, 1 subnets
O   1.1.1.1 [110/2] via 10.10.2.1, 00:28:17, FastEthernet0/0
2.0.0.0/32 is subnetted, 1 subnets
O   2.2.2.2 [110/2] via 10.10.4.1, 00:17:32, FastEthernet1/0
3.0.0.0/32 is subnetted, 1 subnets
C   3.3.3.3 is directly connected, Loopback0
4.0.0.0/32 is subnetted, 1 subnets
O   4.4.4.4 [110/3] via 10.10.4.1, 00:17:06, FastEthernet1/0
10.0.0.0/8 is variably subnetted, 7 subnets, 2 masks
O   10.10.1.0/30 [110/2] via 10.10.4.1, 00:17:32, FastEthernet1/0
    [110/2] via 10.10.2.1, 00:20:22, FastEthernet0/0
C   10.10.2.0/30 is directly connected, FastEthernet0/0
L   10.10.2.2/32 is directly connected, FastEthernet0/0
O   10.10.3.0/30 [110/3] via 10.10.4.1, 00:17:06, FastEthernet1/0
C   10.10.4.0/30 is directly connected, FastEthernet1/0
L   10.10.4.2/32 is directly connected, FastEthernet1/0
O   10.10.5.0/30 [110/2] via 10.10.4.1, 00:17:06, FastEthernet1/0

```

Figure 3: OSPF protocol configuration on PE1 router.

Source: Authors, (2026).

III.4.2 Controller Configuration

The configuration of the OpenDaylight SDN controller is a critical step for the control and management of the software-defined network. This phase involved preparing the controller's runtime environment, setting the necessary environment variables, and loading key features that enable device interaction and network topology management. Initially, the controller distribution directory was accessed using the command `cd distribution-karaf-0.5.2-Boron-SR2`. Subsequently, essential environment variables such as `JAVA_HOME` and `PATH` were exported to ensure the proper execution of the Java environment. The controller was then started by running the script `./bin/karaf`. This procedure was replicated on the second controller to maintain consistency across the environment. Once inside the OpenDaylight console, additional features essential for controller operation were installed. These included `odl-restconf`, which implements Restconf—a REST-based protocol that facilitates control of YANG resources via HTTP. The `L2Switch` feature was also included to provide Layer 2 functionalities, `mdsal-apidocs` to supply API documentation for the MD-SAL Data Store model, and `DLUX`, which offers a web-based interface for visual topology management and other controller configurations.

III.4.3 VXLAN Configuration

To configure VXLAN technology, a virtual bridge (bridge or virtual switch) named `ovsbr0` was first created using the command `ovs-vsctl addbr ovsbr0`. Subsequently, a port named `vxlان0` was added to the `ovsbr0` bridge with the command `ovs-vsctl add-port ovs-br0 vxlan0 - set interface vxlan0 type=vxlan options:remote_ip=192.168.13.2 options:key=1001`. This command configures the `vxlان0` interface as a VXLAN type, assigning it the remote IP address 192.168.13.2 and the VNI 1001. If the default `br0` bridge, automatically created, initially includes the `eth2` interface to connect the PC's LAN network, this interface is removed from `br0` and added to the `ovs-br0` bridge using the commands `ovs-vsctl del-port br0 eth2` and `ovs-vsctl add-port ovs-br0 eth2`, respectively. Finally, VLAN 10 was assigned to the `eth2` port with the command `ovs-vsctl set port eth2 tag=10`. To verify these configurations, the command `ovs-vsctl show` is used, the output of which is shown in Figure 4.

```

/ # ovs-vsctl show
c4955489-f992-428f-b7b5-920322b84733
    Bridge ovs-br0
        Controller "tcp:192.168.100.2:6633"
        Port vxlan0
            Interface vxlan0
                type: vxlan
                options: {key="1001", remote_ip="192.168.13.2"}
        Port ovs-br0
            Interface ovs-br0
                type: internal
        Port eth2
            tag: 10
            Interface eth2

```

Figure 4: Bridge VXLAN configuration in open vSwitch.

Source: Authors, (2026).

In Figure 4, it can be observed that the ovsbr0 bridge is controlled by a device with the IP address 192.168.100.2. To this end, the Spanning Tree Protocol (STP) was enabled using the command `ovsctl set bridge ovsbr0 stp_enable=true`, preventing loops on the ovsbr0 bridge. Next, the bridge was configured to use version 1.3 of the OpenFlow protocol with the command `ovsctl set bridge ovsbr0 protocols=OpenFlow13`. Finally, the controller for the bridge was set via a TCP connection to the IP address 192.168.100.2 and port 6633 using the command `ovsctl set-controller ovsbr0 tcp:192.168.100.2:6633`. The configuration process for OpenvSwitchN2 was analogous to that of OpenvSwitchN1.

III.4.4 Operate

During this phase, network status monitoring was performed to verify the correct implementation of the previously described technologies. Basic connectivity tests, such as ping commands between virtual PCs, were conducted. Figure 5 shows the successful communication from PC1 to PC2.

```
root@PC1:~# ping 192.168.1.3
PING 192.168.1.3 (192.168.1.3) 56(84) bytes of data:
64 bytes from 192.168.1.3: icmp_seq=1 ttl=64 time=368 ms
64 bytes from 192.168.1.3: icmp_seq=2 ttl=64 time=260 ms
64 bytes from 192.168.1.3: icmp_seq=3 ttl=64 time=132 ms
64 bytes from 192.168.1.3: icmp_seq=4 ttl=64 time=247 ms
64 bytes from 192.168.1.3: icmp_seq=5 ttl=64 time=342 ms
64 bytes from 192.168.1.3: icmp_seq=6 ttl=64 time=182 ms
64 bytes from 192.168.1.3: icmp_seq=7 ttl=64 time=113 ms
```

Figure 5: Ping result from PC1 to PC2.

Source: Authors, (2026).

To observe the control plane management in the OpenDaylight controller, a QEMU virtual machine containing the Firefox browser was used. Within this environment, the network control panel was accessed and an IP address was configured within the same subnet as the SDN controller. Once the connectivity parameters were properly set, access to the OpenDaylight controller was established via a web browser by entering the controller's IP address and port 8181. This redirected to the login page, where the default credentials `admin:admin` were used. Upon accessing the OpenDaylight Web User Interface, the presence of the Open vSwitch device operating under the OpenFlow 1.3 protocol was confirmed. Figure 6 shows the controller's graphical interface, which enables the visualization of the network topology and management of associated resources.

III.4.5 Optimize

Several actions were taken to optimize network performance and security. One of the primary measures involved enhancing the security of the routers. A secure password was configured for privileged EXEC mode. Similarly, a password was set for console line access, and authentication was enabled. For remote access, a password was configured on the VTY lines, with access restricted exclusively to the SSH protocol, and authentication was likewise enforced. Additionally, the system domain name was defined and RSA cryptographic keys were generated to support secure services such as SSH.

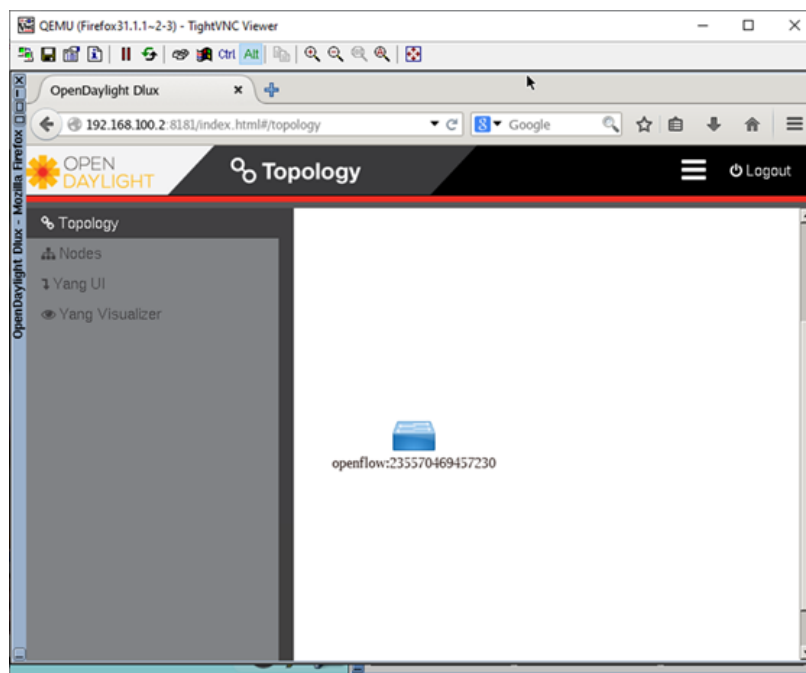


Figure 6: OpenFlow-Managed OVS in OpenDaylight.

Source: Authors, (2026).

Furthermore, specific flow rules were implemented on the ovs-br0 bridges of both Open vSwitch instances to manage traffic behavior. The first rule assigns a priority of 100 to incoming traffic on the eth2 port, forwarding it to the vxlan0 port. The second rule handles traffic entering via the vxlan0 port and forwards it through eth2. Finally, the ovs-br0 bridge was set to secure mode, ensuring it remains operational only when the controller is active. These flow optimizations are illustrated in Figure 7.

```
# ovs-ofctl -O OpenFlow13 add-flow ovs-br0 priority=100,in_port=eth2,actions=output:vxlan0
# ovs-ofctl -O OpenFlow13 add-flow ovs-br0 priority=100,in_port=vxlan0,actions=output:eth2
# ovs-vsctl set-fail-mode ovs-br0 secure
```

Figure 7: Flow rule optimizations on Open vSwitch virtual switches.

Source: Authors, (2026).

Another optimization action consisted of increasing the Maximum Transmission Unit (MTU) on all network interfaces that handle data traffic. The MTU was set to 1530 bytes across all routers to improve packet handling efficiency. Finally, based on preliminary results from the Operate phase, which showed high latency, further optimization was achieved by migrating the emulated network design from a server with 5 GB of RAM to a higher-performance GNS3 server. The new server features 50 GB of RAM and 2 GB of swap space. Monitoring tools confirmed improvements in CPU, memory, and swap usage, as well as system load averages during the execution of the emulation project. This infrastructure upgrade was essential to improve round-trip times for data packets and overall system responsiveness.

IV. RESULTS AND DISCUSSIONS

IV.1 PACKET TRANSMISSION STATISTICS

Wireshark [31] was used to capture and analyze the packets transmitted between the hosts PC1 and PC2. Figure 8 shows a bar graph depicting the number of packets per second captured over time. The maximum number of packets captured at any single time interval was eight.

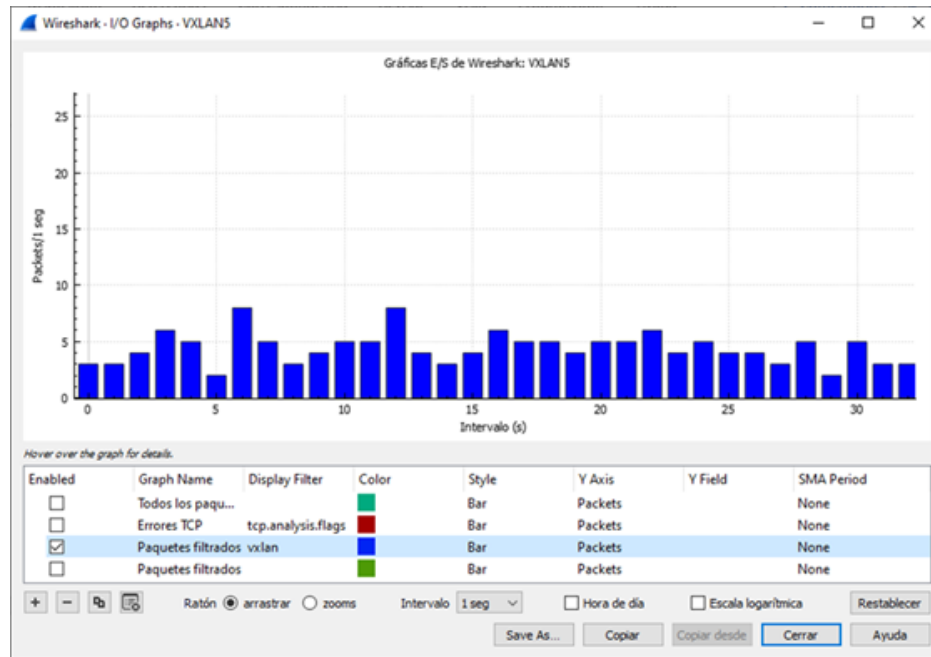


Figure 8: Packet I/O Graph for VXLAN Traffic.

Source: Authors, (2026).

An additional feature of Wireshark is its ability to provide protocol hierarchy statistics. This includes the percentage and total number of packets, the percentage and total number of bytes, bits per second, end packets, end bytes, and end bits per second. Regarding the VXLAN protocol, it accounted for 98.00% of the captured UDP packets, totaling 145 packets during the evaluation period. These packets amounted to 1,168 bytes, with a throughput of approximately 287 bits per second. Furthermore, it is relevant to analyze the statistics at the communication endpoints. Each host PC1 and PC2 recorded a total of 112 packets, 56 sent and 56 received, corresponding to approximately 79,000 bytes in total around 39,000 bytes sent and 39,000 bytes received per host.

IV.2 PERFORMANCE EVALUATION METRICS

The performance evaluation metrics considered in this study include Round Trip Time (RTT) for latency, RTT standard deviation, data transfer volume, and throughput, among others. To measure latency, the ping utility was used to perform three transmission tests from PC1 to PC2. The -O flag was used to prevent packet fragmentation; if the packets exceeded the allowable size, they were not transmitted. The destination IP address was set to 192.168.1.3, and 50 packets were sent using the -c 50 flag. The packet size was set to 1422 bytes using the -s 1422 flag. The aggregated results are presented in Table 4.

Table 4: Ping test results from PC1 to PC2.

Test	RTT (ms)			
	Min	Avg	Max	Std. Dev.
2-5				
1	43.27	64.53	137.60	13.36
2	51.82	61.23	83.38	7.26
3	52.87	60.87	73.30	4.68
4	30.50	50.75	90.20	12.45
5	20.45	30.12	45.67	5.14
6	35.78	70.89	200.23	22.47
7	60.33	75.56	100.45	10.88
8	25.67	40.78	60.12	15.33
9	40.10	55.25	80.45	8.12
10	55.44	68.77	90.56	12.19

Source: Authors, (2026).

The results presented in Table 4 indicate that, in the most representative tests, the average round-trip time (RTT) between PC1 and PC2 ranged from 60.87 ms to 64.53 ms, with a corresponding standard deviation between 4.68 ms and 13.36 ms. These values reflect a typical latency pattern under stable conditions. However, some test cases exhibited greater variability, with RTT averages as low as 30.12 ms and as high as 75.56 ms, underscoring the sensitivity of performance to varying network conditions and system load. Throughput was evaluated using iperf3 with ten parallel UDP streams (-P 10 -u) over a 15-second interval. As shown in Figure 9, the transmission achieved approximately 18.8 MB of total data sent, with an aggregate throughput of about 1.05 Mbit/s. This result reflects the combined impact of VXLAN encapsulation overhead, virtual switch processing, and the intrinsic performance constraints of the emulated environment. While the test is not intended to serve as a high-performance benchmark, it confirms that the SDN-VXLAN overlay sustained stable operation under multiple simultaneous UDP flows, demonstrating functional correctness rather than raw throughput capacity.

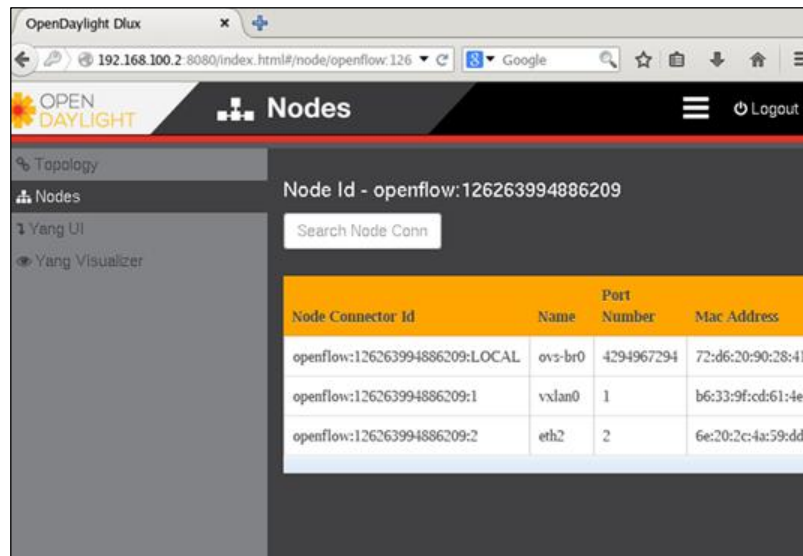
Test	Time	Duration	sec	MBytes	Mbits/sec	Kbytes	Kbits/sec	ms	%
[9]	0.00-15.00	sec	1.88	1.88	1.05	24.6	10.1	0.000	0/1407 (0%)
[9]	0.00-19.85	sec	24.6	24.6	10.1	393.908	805/823 (98%)	393.908	805/823 (98%)
[11]	0.00-15.00	sec	1.88	1.88	1.05	21.8	9.01	0.000	0/1407 (0%)
[11]	0.00-19.85	sec	21.8	21.8	9.01	437.224	807/823 (98%)	437.224	807/823 (98%)
[13]	0.00-15.00	sec	1.88	1.88	1.05	21.8	9.01	0.000	0/1407 (0%)
[13]	0.00-19.85	sec	21.8	21.8	9.01	437.307	807/823 (98%)	437.307	807/823 (98%)
[15]	0.00-15.00	sec	1.88	1.88	1.05	20.5	8.45	0.000	0/1407 (0%)
[15]	0.00-19.85	sec	20.5	20.5	8.45	466.468	807/822 (98%)	466.468	807/822 (98%)
[17]	0.00-15.00	sec	1.88	1.88	1.05	21.8	9.01	0.000	0/1407 (0%)
[17]	0.00-19.85	sec	21.8	21.8	9.01	437.572	806/822 (98%)	437.572	806/822 (98%)
[19]	0.00-15.00	sec	1.88	1.88	1.05	21.8	9.01	0.000	0/1407 (0%)
[19]	0.00-19.85	sec	21.8	21.8	9.01	416.100	806/822 (98%)	416.100	806/822 (98%)
[21]	0.00-15.00	sec	1.88	1.88	1.05	21.8	9.01	0.000	0/1407 (0%)
[21]	0.00-19.85	sec	21.8	21.8	9.01	416.087	806/822 (98%)	416.087	806/822 (98%)
[23]	0.00-15.00	sec	1.88	1.88	1.05	21.8	9.01	0.000	0/1407 (0%)
[23]	0.00-19.85	sec	21.8	21.8	9.01	415.478	806/822 (98%)	415.478	806/822 (98%)
[SUM]	0.00-15.00	sec	18.8	18.8	10.5	503.994	8059/8225 (57%)	503.994	8059/8225 (57%)

Figure 9: Performance Metrics for UDP Transmission using iPerf3.

Source: Authors, (2026).

IV.3 SECURITY AND SCALABILITY

In networks configured with VXLAN technology, vulnerabilities typically arise only from within the LAN, as access from the outside is generally restricted. It is well known that VXLAN networks operate as overlays on existing infrastructure. Considering that VLANs were configured in the implementation, the VTEPs transmit VXLAN traffic over these VLANs, establishing a fundamental security mechanism. The logical bridges ovs-br0 in the Open vSwitch virtual switches are configured in secure mode, ensuring that traffic flows only when the SDN controller is active. Additionally, these bridges segment network traffic according to independent flow rules. As initially mentioned, VXLAN supports up to 16 million tunnels, allowing the configuration of an equivalent number of VNIs. The details of the UDP packets captured with Wireshark show a VNI of 1001, which corresponds to a 24-bit value, confirming the theoretical scalability of the technology. This demonstrates that the VXLAN implementation is highly scalable. The SDN controller enables centralized network management, which greatly facilitates scalability. Devices implementing the OpenFlow protocol are monitored through statistics visualization, as shown in Figure 10. In this figure, it can be observed that the node (logical bridge ovs-br0 of Open vSwitchN1) has three monitored ports: the local port ovs-br0, the vxlan0 port through which encapsulated VXLAN traffic flows, and the physical port eth2, which connects to the PC or host.



Node Connector Id	Name	Port Number	Mac Address
openflow:126263994886209:LOCAL	ovs-br0	4294967294	72:d6:20:90:28:41
openflow:126263994886209:1	vxlan0	1	b6:33:9f:cd:61:4e
openflow:126263994886209:2	eth2	2	6e:20:2c:4a:59:dd

Figure 10: Scalability of an SDN-based Network.
Source: Authors, (2026).

IV.4 DISCUSSION

The experimental results obtained in this study validate the feasibility and correctness of the proposed SDN–VXLAN enterprise architecture while providing empirical evidence that complements the existing body of research reviewed in the related works section. Unlike prior studies, which emphasized performance optimization [14], energy efficiency in multisite datacenters [15], [16], or sector-specific deployments in education and agriculture [17], [18], [23], the present work focuses on delivering a replicable, controller-agnostic design methodology guided by the PPDIOO framework. The obtained measurements confirm that the implemented overlay operates consistently under controlled emulation, with VXLAN encapsulation representing 98% of UDP traffic and symmetrical packet exchange between endpoints, indicating correct tunnel formation and stable forwarding behavior across Open vSwitch VTEPs. Latency and throughput tests reveal performance patterns aligned with expectations for virtualized environments. The RTT values, generally ranging between 60-65 ms in representative tests, reflect the additional encapsulation overhead intrinsic to VXLAN as well as the processing introduced by the SDN controller. Variability observed in certain test iterations is consistent with resource contention and timing fluctuations typical of GNS3/Mininet environments and does not indicate architectural deficiencies. The throughput of 1.05 Mbit/s under 10 parallel UDP streams further confirms that the SDN controller effectively manages flow rule installation while preserving stability during concurrent transmissions. Although not intended as a high-performance benchmark, these results demonstrate that the architecture can support realistic enterprise traffic patterns.

The security and scalability analysis reinforces the suitability of the design for enterprise environments. The combination of VLAN-based segmentation in the underlay and secure Open vSwitch configurations in the overlay ensures controlled forwarding that depends on the SDN controller, limiting the attack surface to internal domains. Furthermore, the use of VNI 1001 and VXLAN's 24-bit addressing space exemplifies the scalability potential of the implementation, enabling organizations to create a large number of isolated virtual networks. Centralized management through the SDN controller provides clear operational benefits, including unified monitoring, streamlined provisioning, and improved visibility of VXLAN tunnel endpoints, all essential characteristics for modern enterprise networks adopting software-defined paradigms. When contrasted with existing literature, the results obtained here highlight the distinct contribution of this work: while previous studies provide insights into specific optimizations, deployments, or sectoral applications, none offer a complete and reusable blueprint for SDN–VXLAN adoption in enterprise settings. The proposed architecture, built on open-source tools Ryu and OVS, reproducible configurations, and a methodical PPDIOO-driven workflow, fills this methodological gap and provides organizations with a practical pathway for transitioning toward scalable and virtualized overlay networks.

V. CONCLUSIONS

This research presented the design and functional validation of an SDN-based VXLAN architecture tailored for enterprise networks, integrating Software-Defined Networking principles with Virtual Extensible LAN overlay mechanisms under a structured PPDIOO lifecycle framework. The implementation in a GNS3 environment using OpenDaylight as controller and Open vSwitch as the switching fabric demonstrated the operational feasibility of centralized VXLAN tunnel orchestration. Functional validation confirmed correct encapsulated traffic exchange, dynamic flow rule installation through OpenFlow, and real-time topology management, supporting the architecture's capability to deliver scalable segmentation and centralized programmability in enterprise contexts. Beyond functional verification, the study contributes a structured methodological approach for implementing software-defined overlay networks by systematically applying the PPDIOO model across all lifecycle phases. This framework enables explicit evaluation of architectural decisions from preparation through optimization, offering a reproducible blueprint for academic and industrial deployments while reinforcing disciplined network design practices in SDN environments. Overall, the proposed architecture constitutes a feasible and replicable foundation for next-generation enterprise networking. Future research will extend validation to real or hybrid infrastructures under production-scale traffic conditions and incorporate advanced security, monitoring, and interoperability mechanisms, including integration with SD-WAN paradigms.

VI. AUTHOR'S CONTRIBUTION

Conceptualization: Daniel Nuñez-Agurto, Luis Paredes-Alcivar

Methodology: Luis Paredes-Alcivar

Investigation: Daniel Nuñez-Agurto, Luis Paredes-Alcivar

Discussion of results: John Cruz-Garzon, Franklin Perez

Writing – Original Draft: Daniel Nuñez-Agurto

Writing – Review and Editing: Daniel Nuñez-Agurto, Franklin Perez.

Resources: John Cruz-Garzon, Franklin Perez.

Supervision: Daniel Nuñez-Agurto, John Cruz-Garzon .

Approval of the final text: Daniel Nuñez-Agurto, Luis Paredes-Alcivar, John Cruz-Garzon, Franklin Perez.

VII. ACKNOWLEDGMENTS

The authors would like to thank the Distributed Systems, Cybersecurity, and Content Research Group (RACKLY) at Universidad de las Fuerzas Armadas - ESPE for its scientific and collaborative contribution.

VIII. REFERENCES

- [1] Cisco, "Cisco annual internet report (2018–2023)," Cisco, 2018–2023. Available: <https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf>
- [2] E. F. Naranjo and G. D. Salazar Ch, "Underlay and overlay networks: The approach to solve addressing and segmentation problems in the new networking era: VXLAN encapsulation with cisco and open source networks," in 2017 IEEE second ecuador technical chapters meeting (ETCM), 2017, pp. 1–6. doi: 10.1109/ETCM.2017.8247505.
- [3] Z. Zhai, Q. Li, et al., "Application discussion of SDN technology in multi-data center," Smart Systems and Green Energy, vol. 2, no. 1, pp. 11–19, 2020.
- [4] M. S. Mulya Sudrajat, D. Perdana, and R. M. Negara, "Performance analysis of VXLAN and NVGRE tunneling protocol on virtual network," Bulletin of Electrical Engineering and Informatics, vol. 6, no. 3, pp. 295–300, Sep. 2017.
- [5] T. Singh, V. Jain, and G. S. Babu, "VXLAN and EVPN for data center network transformation," in 2017 8th international conference on computing, communication and networking technologies (ICCCNT), 2017, pp. 1–6. doi: 10.1109/ICCCNT.2017.8203947.
- [6] D. Nuñez-Agurto, W. Fuertes, L. Marrone, E. Benavides-Astudillo, and M. Vásquez Bermúdez, "Traffic classification in software-defined networking by employing deep learning techniques: A systematic literature review," in Technologies and innovation, Cham: Springer Nature Switzerland, 2023, pp. 67–80. doi: 10.1007/978-3-031-45682-4_6.
- [7] Y. Yan and H. Wang, "Open vSwitch vxlan performance acceleration in cloud computing data center," in 2016 5th international conference on computer science and network technology (ICCSNT), 2016, pp. 567–571. doi: 10.1109/ICCSNT.2016.8070222.
- [8] D. Nuñez-Agurto, W. Fuertes, L. Marrone, M. Castillo-Camacho, E. Benavides-Astudillo, and F. Perez, "Attack classification using machine learning techniques in software-defined networking," in International conference on applied technologies, Cham: Springer Nature Switzerland, 2024, pp. 241–256. doi: 10.1007/978-3-031-58953-9_19.
- [9] D. Nuñez-Agurto, W. Fuertes, L. Marrone, L. Pisco, X. Ordoñez, and G. Rodriguez, "Application-based traffic classification by employing machine learning models on software-defined networking," in International conference on computer science, electronics and industrial engineering, Cham: Springer Nature Switzerland, 2024, pp. 355–370. doi: 10.1007/978-3-031-70981-4_24.
- [10] M. Mahalingam et al., "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks." in Request for comments. RFC 7348; RFC Editor, Aug. 2014. doi: 10.17487/RFC7348.
- [11] D. A. S. GEORGE and A. H. George, "A brief overview of VXLAN EVPN," Ijireeiceinternational Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering, vol. 9, no. 7, pp. 1–12, 2021.
- [12] Z. Zhao, F. Hong, and R. Li, "SDN based VxLAN optimization in cloud computing networks," IEEE Access, vol. 5, pp. 23312–23319, 2017.
- [13] G. Zakurdaev, M. Ismail, and C.-H. Lung, "Dynamic on-demand virtual extensible LAN tunnels via software-defined wide area networks," in 2022 IEEE 12th annual computing and communication workshop and conference (CCWC), 2022, pp. 0995–1000. doi: 10.1109/CCWC54503.2022.9720797.
- [14] Y.-C. Yang, Z.-Y. Jin, L.-H. Yen, and C.-C. Tseng, "SDN-enabled EVPN-VXLAN with P4 accelerated user plane." 2025. doi: 10.23919/APNOMS67058.2025.11181298.
- [15] J. R. Ngo Bilong, M. Ba, A. M. Kachallah, S. Ouya, and C. H. Kane, "Eco-efficient SDN-VXLAN architecture for multi-tenant datacenters: Isolation, automation and resource optimization." pp. 567–573, 2025. doi: 10.1109/ICSCDS65426.2025.11167267.
- [16] A. Dayf, M. El-Aasser, and T. Elshabrawy, "Software-defined networking (SDN) optimizations for green multisite data centers (GMDC)." pp. 89–93, 2025. doi: 10.1109/ICMIS165108.2025.11115544.
- [17] S. Coulibaly, M. Ba, E. G. Gbetie, A. D. Kora, and S. Ouya, "Optimisation of the optical fibre infrastructure in mali for distance learning: An approach based on VxLAN and SDN." 2024. doi: 10.1109/ICNGN63705.2024.10871331.
- [18] E. Ndassimba, E. B. S. Zabolo, and Y. P. K.-S. Dodoagnen, "VXLAN and SDN for smart agriculture in online learning: Case study at higher institute of rural development, mbaïki, university of bangui." pp. 375–380, 2024. doi: 10.1109/MIPRO60963.2024.10569412.
- [19] D. Hou, "Analysis of IPv6 deployment scheme based on SDN architecture." 2024. doi: 10.1109/ICDCECE60827.2024.10548362.
- [20] R. Liu, Z. Chen, Y. Li, X. Zhang, Y. Wang, and B. Han, "Research on the application of white box network equipment in improving the cooperative ability of computer cloud network." pp. 771–775, 2022. doi: 10.1145/3558819.3565190.

- [21] Z. Wenhua, W. M. Hazwan Azamuddin, A. H. Mohd Aman, H. Sallehudin, and Z. Yanke, "Simulation cycle for network scenario based on AD-campus solution." 2022. doi: 10.1109/ICCR56254.2022.9995812.
- [22] E. A. Alvarado-Unamuno and J. E. Arizaga-Gamboa, "Design and implementation of traffic balancer over overlay networks with vxlan tunneling," *Communications in Computer and Information Science*, vol. 1460 CCIS, pp. 125–139, 2021, doi: 10.1007/978-3-030-88262-4_9.
- [23] A. M. Kachallah, A. Seck, T. Kondengar, I. Dioum, and S. Ouya, "Building an hybrid distance learning infrastructure for higher education: Combination of internet and an overlay VxLAN network controlled by floodlight," in *2024 IEEE world engineering education conference (EDUNINE)*, 2024, pp. 1–5. doi: 10.1109/EDUNINE60625.2024.10500644.
- [24] A. S. Elrashdi, S. K. Alferjani, R. R. Omar, and F. M. Hasan, "The efficiency of using PPDIIO methodology to design graduation projects for network department students," in *2024 IEEE 7th international conference on advanced technologies, signal and image processing (ATSIP)*, 2024, pp. 438–442. doi: 10.1109/ATSIP62566.2024.10638951.
- [25] GNS3, "Open vSwitch Appliance for GNS3." 2024. Accessed: Oct. 26, 2024. [Online]. Available: <https://www.gns3.com/marketplace/appliances/open-vswitch>
- [26] I. Amal, A. Fiade, H. Nanang, and Arini, "Performance measurement of first hop redundancy protocol with etherchannel for file transfer services," in *2022 10th international conference on cyber and IT service management (CITSM)*, 2022, pp. 01–07. doi: 10.1109/CITSM56380.2022.9935955.
- [27] F.-Y. Lee, C.-H. Shao, C.-H. Yang, and M.-H. Tsai, "OVS-DPDK-i: A CPU-efficient high-speed open vSwitch," in *2024 international computer symposium (ICS)*, 2024, pp. 145–151. doi: 10.1109/ICS64339.2024.00033.
- [28] B. Pfaff et al., "The design and implementation of open vSwitch," in *12th USENIX symposium on networked systems design and implementation (NSDI 15)*, Oakland, CA: USENIX Association, May 2015, pp. 117–130. Available: <https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/pfaff>
- [29] L. Yin, J. Luo, and K. Li, "An optimal image storage strategy for container-based edge computing in smart factory," *IEEE Internet of Things Journal*, vol. 10, no. 8, pp. 7204–7214, 2023, doi: 10.1109/JIOT.2022.3229206.
- [30] S. Badotra and J. Singh, "Open daylight as a controller for software defined networking." *International Journal of Advanced Research in Computer Science*, vol. 8, no. 5, 2017.
- [31] S. Chowhan and A. K. Saxena, "Advanced techniques in network traffic analysis: Utilizing wireshark for in-depth live data packet inspection and information capture," in *2023 international conference on communication, security and artificial intelligence (ICCSAI)*, 2023, pp. 843–847. doi: 10.1109/ICCSAI59793.2023.10421631.