PROYECTO DE TITULACIÓN

Carrera de Ingeniería en Tecnologías de la Información

TEMA: Diseño de una Red Definida por Software con la tecnología VxLAN para optimizar el rendimiento de las redes empresariales

AUTOR: Paredez Alcívar, Luis Ángel

TUTOR: Ing. Núñez Agurto, Alberto Daniel, Mgtr.

Santo Domingo, 1 de marzo del 2024

Reporte de verificación de contenido

Copyleaks

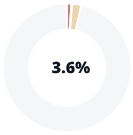
Plagiarism and AI Content Detection Report

SDN_ParedezLuis_Tesis.pdf

Scan details

Scan time:	Total Pages:	Total Words:
March 4th, 2024 at 12:13 UTC	64	15780

Plagiarism Detection



Types of plagiarism		Words
Identical	0.9%	147
Minor Changes	0.3%	52
Paraphrased	2%	311
Omitted Words	9.1%	1442

AI Content Detection

Text coverage		Words
AI text	0%	0
Human text	100%	14338

Learn more

Alerts: (1)

Cross Language: Same Document Language

Submitted language and cross-language text are the same language. No credits were used.

2/5 Severity

Plagiarism Results: (3)

Your File — 2.8%

en el ámbito de las Tecnologías de la Información.

Descripción de EVPN con encapsulación del plano de datos VXLAN | Juno... — 0.6%

https://www.juniper.net/documentation/mx/es/software/junos/evpn-vxlan/topics/concept/evpn-vxlan-data-pl...

X Help us improve your experience. Let us know what you think. ...

2580.pdf — 0.1%

https://www.colibri.udelar.edu.uy/jspui/bitstream/20.500.12008/19026/1/2580.pdf

Software Defined Networking en redes Inalámbricas Ari Chamlian1 , Gastón Telfeyan1 y Nicolás Piquerez1 1 Facultad de Ingeniería, Univers...

Departamento de Ciencias de la Computación

Carrera de Ingeniería en Tecnologías de la Información

Certificación

Certifico que el trabajo de integración curricular: "**Diseño de una Red Definida por Software con la tecnología VxLAN para optimizar el rendimiento de las redes empresariales**" fue realizado por el señor **Paredez Alcívar, Luis Ángel,** el mismo que cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, además fue revisado y analizada en su totalidad por la herramienta de prevención y/o verificación de similitud de contenidos; razón por la cual me permito acreditar y autorizar para que se lo sustente públicamente.

Santo Domingo, 1 de marzo del 2024

Firma:

ALBERTO DANIEL NUNEZ AGURTO

.....

Núñez Agurto, Alberto Daniel

C. C: 1716572548

Departamento de Ciencias de la Computación

Carrera de Ingeniería en Tecnologías de la Información

Responsabilidad de Autoría

Yo, **Paredez Alcívar, Luis Ángel**, con cédula de ciudadanía n° 2351050303, declaro/declaramos que el contenido, ideas y criterios del trabajo de integración curricular: **Diseño de una Red Definida por Software con la tecnología VxLAN para optimizar el rendimiento de las redes empresariales** es de mi autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Santo Domingo, 1 de marzo del 2024

Firma:

.....

Paredez Alcívar, Luis Ángel

C.C.: 2351050303

Departamento de Ciencias de la Computación

Carrera de Ingeniería en Tecnologías de la Información

Autorización de Publicación

Yo **Paredez Alcívar, Luis Ángel**, con cédula de ciudadanía n° 2351050303, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de integración curricular: **Diseño de una Red Definida por Software con la tecnología VxLAN para optimizar el rendimiento de las redes empresariales** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Santo Domingo, 1 de marzo del 2024

Firma:

.....

Paredez Alcívar, Luis Ángel

C.C.: 2351050303

DEDICATORIA

Dedico este proyecto de titulación, y en general, toda mi carrera universitaria, a mis padres, Ángel Paredez y Francisca Alcívar, por su esfuerzo y sacrificio para facilitarme todos los recursos para obtener un nuevo título académico.

A mis hermanos, Diego, Tatiana y Melissa, por su apoyo incondicional, que fue clave para mi formación universitaria y el cumplimiento de mis objetivos académicos.

Paredez Alcívar, Luis Ángel

AGRADECIMIENTO

Agradezco a mis padres y mis hermanos por ser parte esencial de mi formación académica. El cumplimiento de mis metas y objetivos alcanzados se deben al apoyo constante de ellos.

Mi sincero agradecimiento al Ing. Daniel Núñez, quien fue el tutor del presente proyecto de titulación. Su guía fue fundamental para aclarar dudas y alcanzar los objetivos estipulados. Además, me gustaría reconocer la labor de todos los docentes que fueron partícipes a lo largo de mi carrera universitaria.

Por último, agradezco a la Universidad de las Fuerzas Armadas ESPE por brindarme la infraestructura para desarrollar mis habilidades académicas y contribuir a mi formación profesional.

Paredez Alcívar, Luis Ángel

ÍNDICE

Dedicatoria I

Agradecimiento II

Resumen 1

Abstract 2

I Introducción y estado del arte 3

A Introducción 3

B Estado del arte: 4

1 Planificación de la revisión 6

2 Criterios de construcción de la cadena 7

3 Elaboración del estado del arte 8

C Objetivos 8

1 Objetivo General 8

2 Objetivos Específicos 8

D Alcance 9

II Marco teórico/Marco conceptual 10

A Redes Definidas por Software 10

1 Arquitectura SDN 10

2 OpenFlow 12

3 Arquitectura OpenFlow 12

4 Controladores SDN 12

B VxLAN 14
C Multiprotocol Label Switching 16
D Métricas de rendimiento 17
E Metodología PPDIOO 17

III Metodología/Técnicas/Diseño 18
A Metodología de Desarrollo 18
1 Preparación 18
2 Planificación 19
3 Diseño: 20
4 Implementación: 22
5 Operación: 35
6 Optimización: 37

IV Resultados 40
A Estadísticas de transmisión de paquetes 40
B Métricas de evaluación de rendimiento 43
C Seguridad y escalabiliad 47

V Conclusiones y recomendaciones 49
A Conclusiones 49
B Recomendaciones 50
C Trabajos futuros 50

VI Referencias 52

ÍNDICE DE TABLAS

I CONTROLADORES SDN DE CÓDIGO ABIERTO. 14

II PLAN DE TRABAJO PARA EL DISEÑO DE UNA RED DEFINIDA POR SOFTWARE CON TECNOLOGÍA VXLAN PARA LA OPTIMIZACIÓN DE REDES EMPRESARIALES. 20

III DIRECCIONAMIENTO IP DE LOS DISPOSITIVOS. 20

IV RESULTADOS DE TRES PRUEBAS DE PING DESDE LA PC1 A LA PC2. . . . 44

ÍNDICE DE FIGURAS

1 Arquitectura de SDN [17]. 11

2 Arquitectura OpenFlow [20]. 12

3 Formato de paquete VxLAN encapsulado [28]. 15

4 Topología de red de prueba 22

5 Asignación de direcciones IP en el router P1 22

6 Enrutamiento del router P1 23

7 Configuración de MPLS en P1 24

8 Rango de etiquetas mpls en P1 24

9 Asignación de direcciones IP en el router P2 24

10 Enrutamiento del router P2 25

11 Rango de etiquetas mpls en router P2 25

12 Asignación de direcciones IP en el router PE1. 25

13 Configuración de VRF CLIENTEA en el router PE1 25

14 Configuración del protocolo OSPF en el router PE1 26

15 Configuración inicial del protocolo BGP en el router PE1 27

16 Configuración final del protocolo BGP en router PE1 27

17 Configuración de MPLS en el router PE1 28

18 Asignación de direcciones IP en el router PE2 28

19 Configuración de VRF CLIENTEA en PE2 28

20 Configuración del protocolo OSPF en el router PE2 29

21 Configuración final del protocolo BGP en router PE2 29

22 Configuración de MPLS en el router PE2 29

23 Asignación de direcciones IP en el router CE1 30

24 Configuración de BGP en router CE1 30
25 Asignación de direcciones IP en el router CE2 30
26 Configuración de BGP en router CE2 31
27 Asignación de direcciones IP en el OpenvSwitchN1 31
28 Configuración del puente ovs-br0 para establecer VxLAN en OpenvSwitchN1 . . . 32
29 Asignación de direcciones IP en el OpenvSwitchN2 33
30 Configuración del puente ovs-br0 para establecer VxLAN en OpenvSwitchN1 . . . 33
31 Iniciación de OpenDaylight 33
32 Ejecución de OpenDaylight 34
33 Interfaz eth0 de la PC1 34
34 Interfaz eth0 de la PC2 34
35 Ping de PC1 a PC2 35
36 Definición de parámetros de conexión en imagen QEMU Firefox 35
37 Inicio de sesión en OpenDaylight 36
38 Visualización de dispositivo con el protocolo OpenFlow 36
39 Configuraciones de seguridad en routers 37
40 Optimizaciones de reglas de flujo en los switches virtuales Open vSwitch 38
41 Optimización de mtu en router 38
42 Optimización de mtu en switch virtual Open vSwitch 38
43 Características del servidor GNS3 39
44 Captura de paquetes en Wireshark 40
45 Detalle de paquete capturado en Wireshark 41
46 Gráfico E/S de paquetes VxLAN 41
47 Estadísticas de Jerarquía de Protocolo 42

48 Estadísticas de Nodos con el protocolo IPv4 43
49 Estadísticas de Nodos con el protocolo UDP 44
50 Estadísticas de Nodos con el protocolo TCP 45
51 Resultados de prueba de ping de PC1 a PC2 45
52 Servidor de herramienta IPERF3 46
53 Métricas de evaluación con el protocolo UDP 46
54 Métricas de evaluación con el protocolo TCP 47
55 Escalabilidad de VxLAN 48
56 Escalabilidad de una red SDN 48

RESUMEN

En la actualidad, las tecnologías de virtualización son parte fundamental de las redes empresariales. La tecnología de Red de Área Local Virtual Extensible (VxLAN) permite encapsular el tráfico de red de capa 2 en paquetes UDP para su transporte sobre la capa 3. Esto hace que sea utilizado en entornos donde se necesitan redes flexibles y escalables. La virtualización tecnológica en los centros de datos está estrechamente ligada con las Redes Definidas por Software (SDN), que contribuyen a la separación del plano de control y de datos. El presente trabajo tiene como objetivo diseñar e implementar una Red Definida por Software utilizando la tecnología VxLAN, para mejorar el rendimiento y la eficiencia de las redes empresariales. Se necesitó realizar una investigación sobre artículos científicos que implementen estas tecnologías. Para llevar a cabo el trabajo, se empleó la metodología PPDIOO. El diseño se implementó mediante el software emulador GNS3. Se requirió del software controlador de SDN OpenDaylight, así como el switch virtual de código abierto Open vSwitch configurados en las plantillas Ubuntu Docker Guest de GNS3. En ellos se configuró la red superpuesta VxLAN. La interconexión se realizó mediante la red subyacente MPLS utilizando routers, en los cuales se configuró los protocolos BGP Y OSPF. Los resultados obtenidos demuestran que el diseño propuesto optimiza el rendimiento de una red empresarial. La combinación de la tecnología VxLAN y SDN ofrece mecanismos de seguridad y escalabilidad, aspectos que son fundamentales para las redes empresariales.

Palabras clave: Red Definida por Software, VxLAN, Redes empresariales.

ABSTRACT

Virtualization technologies are now a fundamental part of enterprise networks. Virtual Extensible Local Area Network (VxLAN) technology allows Layer 2 network traffic to be encapsulated in UDP packets for transport over Layer 3. This makes it suitable for use in environments where flexible and scalable networks are required. Technological virtualization in data centers is closely linked with Software Defined Networking (SDN) which contributes to the separation of the control and data plane. The objective of this work is to design and implement a Software Defined Network using VxLAN technology to improve the performance and efficiency of enterprise networks. It was necessary to conduct a research on scientific articles that implement these technologies. To carry out the work, the PPDIOO methodology was used. The design was implemented using the GNS3 emulator software. The OpenDaylight SDN driver software was required, as well as the Open vSwitch open source virtual switch configured in the Ubuntu Docker Guest templates of GNS3. The VxLAN overlay network was configured on them. The interconnection was performed through the underlying MPLS network using routers, on which BGP AND OSPF protocols were configured. The results obtained show that the proposed design optimizes the performance of an enterprise network. The combination of VxLAN and SDN technology provides security and scalability mechanisms, which are essential for enterprise networks.

Keywords: Software-Defined Networking, VxLAN, Enterprise networks.

I. Introducción y estado del arte

A. *Introducción*

Las nuevas herramientas tecnológicas con propósitos comunicacionales concentran grandes ventajas para sobrellevar la escalabilidad y rendimiento de una red empresarial. En este sentido las Redes Definidas por Software (SDN) [1], proporcionan soluciones de gestión y centralización de las redes empresariales mejorando la eficacia de esta. Por otro lado, la tecnología de Red de Área Local Virtual Extensible (VxLAN) [1], permite crear un gran número de redes aisladas virtuales superando las limitaciones de tecnologías como las VLANs. Con ello, cada servidor físico tiene la condición favorable de contener servidores virtuales, cada uno de los cuales cuenta con su propia dirección IP.

La combinación de las redes SDN con la tecnología VxLAN reúne las características necesarias para sobrellevar las limitaciones de las tecnologías actuales. Para alcanzar los resultados esperados, se requiere de un diseño topológico adecuado que utilice los dispositivos necesarios que permitan configurar una red utilizando las tecnologías ya mencionadas. El estudio de los dispositivos virtuales estará supeditado al rendimiento y eficacia que supone el uso de estos en una red empresarial.

Luego de la elección de los dispositivos virtuales, estos se dispondrán en un diseño de red apropiado siguiendo la metodología PPDIOO, posteriormente se configurarán a través de la consola de comandos y/o interfaces gráficas propias de los softwares. Una vez obtenida la conexión de extremo a extremo mediante tecnología VxLAN, se verificarán las estadísticas de la red accediendo al servicio web del controlador SDN y utilizando herramientas de análisis de tráfico de red. Lo que permitirá realizar una gestión apropiada, además se verificará el flujo de datos y la mejora del rendimiento de la red.

El rendimiento es una de las características fundamentales en una red empresarial. La eficiencia de la red es clave para la interconexión de extremo a extremo y flujo de datos ininterrumpido. Las tecnologías comunes no son capaces de hacer frente a la escalabilidad por lo que se necesita de la implementación de nuevas tecnologías. Es en este punto donde se hace factible el diseño de SDNs que en combinación con tecnologías como VxLAN permiten cubrir la demanda de una red.

La administración de una red se vuelve compleja cuando crece el número de dispositivos físicos que la conforman. A diferencia de una red administrada mediante un software que centraliza la gestión y administra las estadísticas propias de una red.

Las topologías tradicionales no son capaces de cumplir eficientemente las exigencias de las nuevas tecnologías, y el crecimiento continuo de los nodos de una red empresarial requiere utilizar arquitecturas modernas que puedan satisfacer las necesidades organizacionales procurando que el nivel de costos sea el mínimo posible. Ante lo cual, las tecnologías de virtualización, sistemas computacionales y sistemas de contenedores son vistas como las mejores opciones.

Por lo tanto, esta investigación busca diseñar una red eficiente SDN, que por medio de la tecnología VxLAN, se pueda optimizar el rendimiento de las redes empresariales. Es necesario tener en cuenta que existen notables diferencias entre las redes tradicionales y el uso de redes SDN, que van desde el nivel de complejidad de administración, hasta la infraestructura física de red.

B. Estado del arte:

El avance significativo de las redes de comunicación y las necesidades en entornos empresariales han marcado el surgimiento de nuevas tecnologías. Las empresas tienen el desafío de aprovechar las oportunidades de cambio y proponer nuevas arquitecturas que suplan las necesidades industriales. En [2], se señala que la virtualización de redes ha sido influyente en la transformación de la red tradicional a una de red definida por software para resolver problemas de expansión de usuarios.

Los investigadores dieron paso a nuevas tecnologías, entre las que figuraban las Redes Definidas por Software. Esta tecnología representan un modo alterno de comunicación a las redes de tradicionales, con la principal característica de separar el plano de datos y el plano de control. En [3], se menciona que la importancia de contar con un proceso de comunicación eficiente.

En [4], se presentan los conceptos principales de una SDN, así como los componentes claves de una infraestructura con estos tipos de redes mediante el enfoque bootom-up y capas. Abordan plataformas de control enfocándose en parámetros como la seguridad, escalabilidad, rendimiento. Señalan cómo las SDN permiten tener una visión global de la red, programación

dinámica en dispositivos que permite una mejor adaptabilidad configuración de la red. Además, la gestión de la red disminuye la complejidad al establecer políticas de seguridad.

En [5], se evalúa diferentes aspectos de las SDN como emuladores y simuladores, plataformas de control, problemas de seguridad, pruebas de red. Recalcan que la seguridad tiene un papel fundamental en la red SDN. Además, los investigadores determinan que las herramientas de simulación son importantes para probar aplicaciones desarrolladas en el ámbito académico e industrial. También, indican que ciertas herramientas de simulación Overlay utilizan conmutadores como Open vSwitch y máquinas virtuales, y para la interconexión de éstas, emplean encapsulación VxLAN.

Dentro de las tecnologías empleadas en las redes empresariales se encuentran las Redes de Área Local Virtuales (VLANs), que permiten segmentar una red y utilizan protocolos como el Spanning Tree para la prevención de bucles. Ante el surgimiento de nuevas tecnologías que cuentan con la función de encapsular el tráfico de red de capa 2 sobre capa 3, se dio paso a VxLAN. En [6], se expone cómo la tecnología VxLAN puede ayudar a las organizaciones a aprovechar el potencial de las redes modernas, indica entre otros aspectos sobre las limitaciones de las VLAN, las cuales son solucionadas en gran parte por VxLAN, entre ellas destacan la escalabilidad en la capa 2 y la conexión por medio de la capa 3.

La implementación de las nuevas tecnologías es producto del crecimiento de arquitecturas, especialmente basadas en la nube, en los centros de datos pretendiendo no afectar el flujo de datos ni la conexión de un extremo al otro. La combinación de las nuevas tecnologías de configuración de red con tecnologías ya conocidas como los entornos de virtualización, así como aplicaciones de código abierto son capaces de reunir las características más importantes de los centros de datos actuales como el rendimiento, escalabilidad y flexibilidad, según lo expone [7]. Esto quiere decir que el continuo crecimiento en los centros de datos que tienen un bajo nivel de modernización, puede ocasionar inconvenientes a nivel de conectividad si no cuentan con la capacidad de operación tecnológica suficiente para cubrir la demanda de conexión.

En [8], se recalca cómo la virtualización ha sido unos de los paradigmas de mayor interés en el área de las redes, cuya aplicación ha tenido un impacto importante en el ámbito de las tecnologías de la información. Se realiza un estudio comparativo entre las tecnologías de virtualización dentro de las cuales se encuentran VxLAN y SDN. Se enfoca en aspectos como:

flexibilidad, seguridad, escalabilidad, gestión, entre otros. Los resultados muestran que ciertas tecnologías pasan por alto algunos aspectos para promover otros. También, señalan la necesidad de explorar las deficiencias y mejorar los entornos de virtualización.

Las SDN en conjunto con VxLAN pueden brindar soluciones en pro de la optimización del rendimiento de las redes empresariales en un contexto de crecimiento y flujo continuo de datos aprovechando el hardware y especialmente el software disponible. El surgimiento de switches virtuales de código abierto como Open vSwitch tiene un impacto positivo en el ahorro de recursos en la adquisición de equipos físicos, sobre todo cuando pueden tener funciones de conectividad al trabajar en múltiples capas. Además, puede ser implementado en sistemas de contenedores e hipervisores entre los cuales destaca Docker [9].

El Open vSwitch es un software que implementa el protocolo OpenFlow, cuya aplicación principal es el soporte de SDN por medio de un controlador. No obstante, Open vSwitch puede soportar otros protocolos de administración de switch. Es necesario señalar que Open vSwitch se puede utilizar como un dispositivo con aplicaciones netamente de switch en sistemas virtualizados y para distintos fines en la conmutación de nodos físicos [10].

Para profundizar el estado del arte, se realizó una revisión sistemática de la literatura acorde a los delineamientos propuestos por [11]. Las investigaciones contienen información acerca de diseños e implementación de Redes Definidas por Software, así como del uso de la tecnología VxLAN en el ámbito investigativo y empresarial.

1. Planificación de la revisión

Identificación de la necesidad de revisión: es necesario identificar el problema principal del presente proyecto, para después llevar a cabo la investigación de los artículos o estudios importantes con el propósito de consolidar el conocimiento sobre los temas tratados. Asimismo, la revisión brinda la oportunidad de conocer el trabajo previo de investigadores del tema en cuestión.

Especificación de las preguntas de investigación: se formularon tres prreguntas de investigación que se responderán conforme se desarrolle la revisión de la literatura.

RQ1. ¿Cómo las SDN contribuyen a la optimización del rendimiento de una red?

RQ2. ¿Cuáles son los beneficios del uso de la tecnología VxLAN en una red empresarial?

RQ3. ¿Cuáles son los aspectos evaluados para medir el rendimiento de una red?

Criterios de inclusión y exclusión: los criterios de inclusión empleados para identificar estudios relacionados con la presente investigación son:

- Trabajos sobre SDN.
- Investigaciones que implementen la tecnología VxLAN.
- Estudios que analicen el rendimiento de las SDNs y VxLAN.

Por otro lado, los criterios de exclusión establecidos son:

- Artículos que no muestren métricas de rendimiento de redes.
- Artículos no escritos en inglés.
- Artículos que no aborden las tecnologías SDN y VxLAN.

2. Criterios de construcción de la cadena

Proceso de búsqueda: para la búsqueda de los artículos que abordan el tema de investigación, se necesitó utilizar las bases de datos de carácter científico IEEE Xplore y Scopus. Dicha búsqueda requirió definir palabras claves, tales como: Software-Defined Networking, Virtual Extensible LAN, performance y optimization. De esta manera, la cadena de búsqueda se estableció de la siguiente forma:

((“Software-Defined Networking” OR “Software Defined Network” OR “SDN”) AND (“Virtual Extensible LAN” OR “VxLAN”) AND (“Performance”) AND (“Optimization”))

Realizar la revisión: se hallaron un total de 36 artículos, de los cuales, el más relevante fue 1 artículo que contenía las palabras clave buscadas y cumplía con los criterios de inclusión. Con la información extraída del artículo se responden las preguntas definidas anteriormente.

Respondiendo a RQ1., las SDN hacen énfasis en la separación del plano de control y el plano de datos. El aprovechamiento que se le puede dar mediante el uso de sus características programables y especialmente de su visión global es clave para mejorar el rendimiento de una red. Además, puede ser capaz de controlar de forma centralizada redes como VxLAN [1].

Respecto a RQ2., la tecnología VxLAN permite tener más de 16 000 000 de usuarios que pueden estar en la misma infraestructura física [1]. Además, al utilizar los switches de borde VTEP para encapsular y desencapsular los mensajes, se optimiza el procesamiento de dichos

mensajes, lo que disminuye la carga en los dispositivos centrales.

Respondiendo a RQ3., algunos de los aspectos que los investigadores de distintos estudios evalúan para medir el rendimiento son: escalabilidad, tráfico de red, seguridad, interoperatibilidad, entre otros. En el intercambio de información y en procesos de migración una de las métricas analizadas es el throughput [1].

3. Elaboración del estado del arte

EP1. SDN Based VxLAN Optimization in Cloud Computing Networks [1]. En este artículo proponen una arquitectura de red VxLAN que se encuentra basada en una Red Definida por Software. Presentan cómo implementarse en un centro inteligente que permita migrar fácilmente máquinas virtuales. Utilizaron Mininet para desplegar la arquitectura y tener resultados óptimos de equilibrio de carga. Concluyen que la arquitectura VxLAN basada en SDN reduce costos de recursos de multidifusión IP, y puede satisfacer las necesidades para ser implementados en redes de computación en la nube.

C. Objetivos

1. Objetivo General

Diseñar e implementar una Red Definida por Software utilizando la tecnología VxLAN para mejorar el rendimiento y la eficiencia de las redes empresariales.

2. Objetivos Específicos

- Investigar la tecnología VxLAN en el contexto de las Redes Definidas por Software, analizando sus fundamentos, características, ventajas y desafíos.
- Realizar el diseño e implementación de una arquitectura de Red Definida por Software en un entorno de emulación utilizando la tecnología VxLAN.
- Evaluar el impacto de la implementación de la arquitectura de Red Definida por Software con la tecnología VxLAN en términos de rendimiento, eficiencia y seguridad de la red.
- Documentar el proceso de implementación y configuración de la red, para la facilitar la replicación del proyecto en el futuro.

D. Alcance

El presente proyecto busca realizar una investigación, diseño, implementación y evaluación de rendimiento de una Red Definida por Software (SDN) en combinación con la tecnología de Red de Área Local Virtual Extensible (VxLAN), utilizando el software de emulación de redes GNS3 y el sistema de contenedores Docker. Con ello se busca mejorar el rendimiento, eficiencia y seguridad de una red empresarial.

II. MARCO TEÓRICO/MARCO CONCEPTUAL

En esta sección, se exponen los conceptos más importantes y necesarios para comprender el desarrollo y funcionamiento de las tecnologías SDN y VxLAN.

A. Redes Definidas por Software

Las Redes Definidas por Software o SDN son conceptualizadas como una tecnología basada en software capaz de separar físicamente el plano de control del plano de datos, con el fin de centralizar el control de una red, así como programar dispositivos por medio de peticiones y respuestas de una Interfaz de Programación de Aplicaciones o API [12]. Mediante la tecnología SDN, se logra mayor flexibilidad al desacoplar el software que administra el enrutamiento de datos del dispositivo real de transmisión de datos. Con ello, se analiza el flujo de datos, se observa el comportamiento de red y, en general, de los dispositivos que forman parte de ella. Las SDN facilitan el análisis de tráfico puesto que actualizan de forma dinámica la tabla de flujos [13].

La comunicación entre el plano de control y el plano de datos se produce mediante protocolos de comunicación, siendo OpenFlow el más destacado. La gestión del plano de control se realiza desde un controlador SDN, mientras que el plano de datos abarca los dispositivos de red como routers, switches, switches virtuales, entre otros. De esta manera, el plano de control gestiona de forma directa el plano de datos [14].

1. Arquitectura SDN

La estructura de una SDN, como se muestra en la Fig. 1, se encuentra conformada por tres capas: plano de aplicación, plano de control y plano de datos. Según [15], el plano de aplicación se conforma de un grupo de programas que son útiles para el funcionamiento del sistema y responde al entorno de la SDN mediante el controlador. La interfaz norte, también conocida como la interfaz NorthBound, es la encargada de la comunicación que se produce entre el plano de aplicación y el controlador.

En el plano de control, centralizado de forma lógica, se encuentra el controlador SDN, considerado el sistema operativo de la red, que se encarga de gestionar las solicitudes desde el plano de datos. La función esencial del controlador es proveer recursos para las aplicaciones, estadísticas, información topológica, entre otros. La comunicación entre controladores en arqui-

tecturas distribuidas se realiza por medio de la interfaz este-oeste. Cada controlador propone una interfaz, ya que no existe una estandarización en este aspecto. De acuerdo con [16], los controladores que tienen una visión global simplifican la inteligencia de la red.

Por otro lado, la interfaz Sur o SouthBound se utiliza para la comunicación entre el plano de datos y el plano de control, haciendo uso de los protocolos OpenFlow, OVSDB, ForCes, OpFlex, NetConf, entre otros. Por su parte, el plano de datos comprende dispositivos de red entre los cuales figuran principalmente los routers y switches. Se encuentra en la parte más baja de la arquitectura SDN [17]. Los dispositivos de este plano cumplen un papel importante para descubrir la topología y tomar las decisiones de reenvío, especialmente basadas en el Servicio de Seguimiento de Hosts y el Servicio de Descubrimiento de Enlaces.

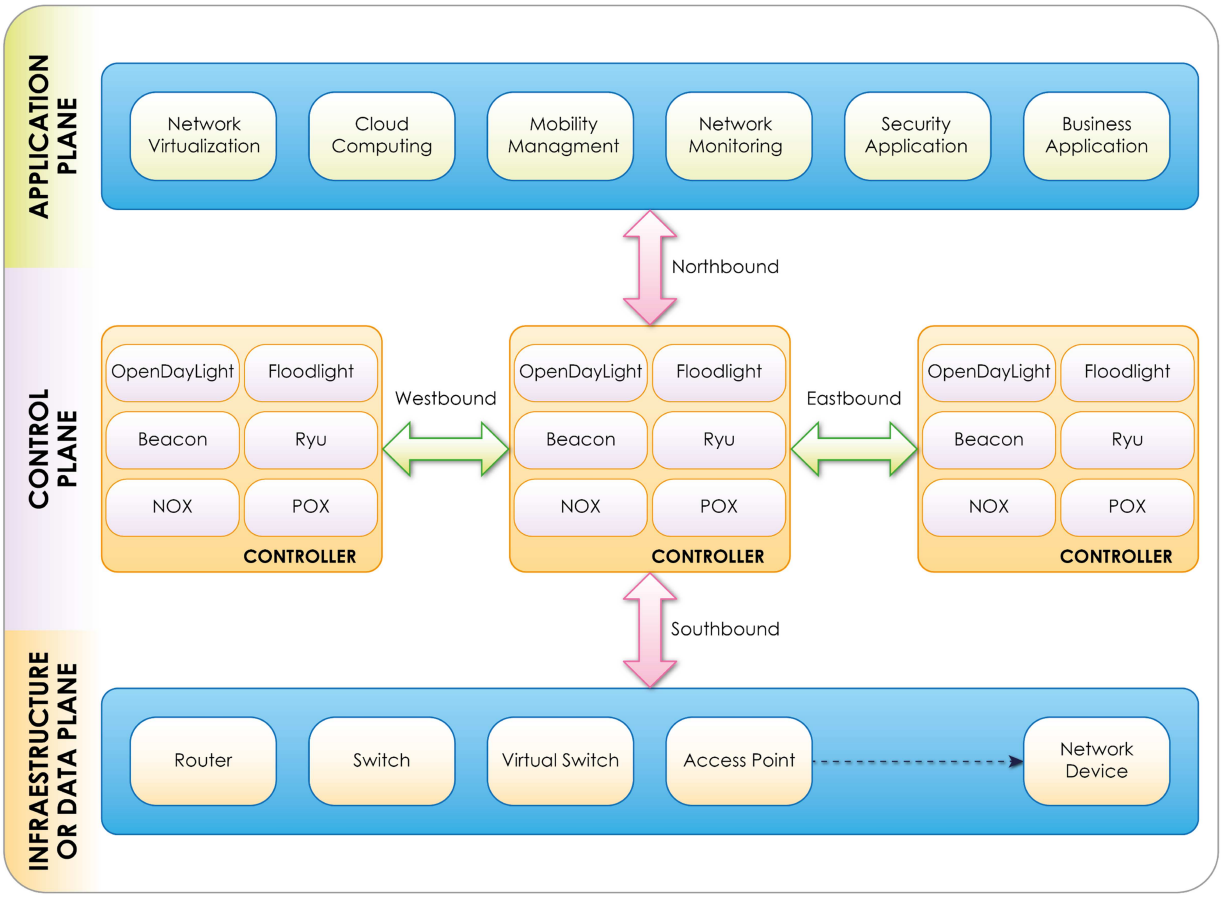


Fig. 1. Arquitectura de SDN [17].

2. *OpenFlow*

En una red SDN se requiere el uso de protocolos que permitan la comunicación entre los controladores y los dispositivos controlados. El protocolo OpenFlow cumple con la función de establecer la comunicación entre, el switch que pertenece al plano de datos, y el controlador, que pertenece al plano de control [18]. La popularidad de OpenFlow creció conforme grandes empresas del sector tecnológico lo implementaron en sus redes principales de centros de datos, brindando soporte en sus productos, especialmente para facilitar la gestión de la nube.

3. *Arquitectura OpenFlow*

La arquitectura OpenFlow como se observa en la Fig. 2, está conformada por tres componentes fundamentales: El protocolo OpenFlow, el dispositivo OpenFlow y el controlador OpenFlow. El protocolo utiliza un canal seguro sobre la Seguridad de la Capa de Transporte y actúa como una interfaz entre los dispositivos OpenFlow y el controlador. El dispositivo OpenFlow puede ser un switch, punto de acceso o router, y en general cualquiera que sea comptatible con OpenFlow. El Controlador sirve para actualizar la tabla flujo mediante la adición o eliminación de entradas de flujo. Estas, a su vez, se componen de un campo regla que sirve para especificar la condición de coincidencia, un campo acción que indica el tipo de acción que se aplicará sobre el flujo, y un campo estadística para contar las veces que ha actuado la regla [19].

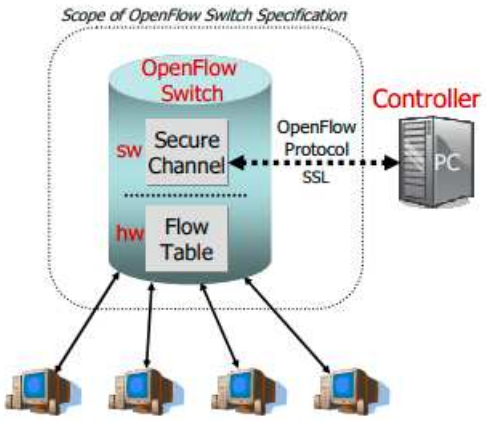


Fig. 2. Arquitectura OpenFlow [20].

4. *Controladores SDN*

Los controladores constituyen el núcleo de una infraestructura SDN al tener una visión completa de la red en la que se incluyen los dispositivos del plano de datos [21]. En este sentido,

las funciones principales del controlador están ligadas al descubrimiento de la topología y al flujo de tráfico. El controlador contiene un módulo que se encarga de descubrir los enlaces, el cual actúa enviando consultas a través de los puertos externos haciendo uso de mensajes de salida de paquetes, los cuales retornan en forma de mensajes de paquetes de entrada, permitiendo al controlador generar la topología de la red.

Existe una amplia gama de controladores, entre los cuales destacan los controladores open source como Floodlight, basado en el controlador Beacon (ambos programados en Java y soportados por los sistemas operativos Linux, MacOS y Windows), entre las características principales de este controlador se encuentran: utiliza bloqueo asincrónico, es modular, cuenta con licencia Apache y compatible con conmutadores OpenFlow, tanto físicos como virtuales. Otro controlador destacado es OpenDaylight, que surgió de la Fundación Linux en el año 2013. Este controlador permite la programación de protocolos como OpenFlow, Netconf, I2RS, entre otros. Entre sus características destacan: se encuentra basado en lenguaje Java, acepta la programación dentro de un entorno bidireccional REST y OSGI con el fin de que aplicaciones se ejecuten al mismo tiempo y espacio de direcciones que el controlador. OpenDaylight está soportado por sistemas operativos Linux, Windows, MacOS, así como por Cisco, Juniper, HP, IBM VMware [22].

Otro controlador de código abierto es Ryu, el cual a diferencia de los anteriores, se encuentra programado en el lenguaje de programación Python y está soportado por las distribuciones de Linux. Entre sus características destacadas se encuentran sus diversos componentes con interfaces de programación que permiten la creación de nuevas aplicaciones y componentes personalizables para la administración de redes organizacionales, adaptándose a sus necesidades.

Por otro lado, está POX, que al igual que Ryu, está basado en Python. Este controlador que está soportado por los sistemas operativos de Windows, Linux y MacOS puede utilizar OpenFlow u OVSDB para establecer comunicación con los switches SDN. La gestión de los componentes de POX puede ocurrir desde la línea de comandos. Gracias a su capacidad para realizar prototipos rápidos, se utiliza especialmente en investigaciones [23].

El controlador NOX desarrollado en el lenguaje de programación C++ y soportado por las distribuciones de Linux, fue desarrollado por Nicira Networks, al igual que OpenFlow; fueron proyectos pioneros de la empresa que contribuyeron a la comunidad de investigadores ofreciendo

una plataforma de control de red con interfaces programáticas. Estas interfaces son esenciales para el desarrollo de nuevas aplicaciones que definen el enrutamiento de cada flujo de red.

Por otra parte, se encuentra ONOS, que, junto con OpenDaylight, constituyen los dos controladores más conocidos. Es programado en Java y soportado por sistemas operativos de Linux, MacOS y Windows. Fue diseñado para redes de operadores de telecomunicaciones puesto que además de sus servicios iniciales brinda nuevos servicios SDN acorde a las necesidades. Una de sus ventajas principales radica en la posibilidad de administrar redes a gran escala, especialmente de alta velocidad, además brinda soporte a redes híbridas [23]. La lista de controladores se resume en la Tabla I.

TABLA I
CONTROLADORES SDN DE CÓDIGO ABIERTO.

Controlador	Lenguaje de programación	Sistema Operativo
Floodlight	Java	Linux, MacOS y Windows
Beacon	Java	Linux, MacOS y Windows
Opendaylight	Java	Linux, MacOS y Windows
Ryu	Python	Linux
POX	Python	Linux, MacOS y Windows
NOX	C++	Linux
ONOS	Java	Linux, MacOS y Windows

B. VxLAN

Existen diversas alternativas para la conexión de extremo a extremo de redes aisladas. Entre las más actuales se encuentra VxLAN, una tecnología que ayuda a sobrepasar los límites que se producen por el aislamiento de redes o subdivisiones de direcciones IP. El protocolo de tunelización VxLAN facilita la subdivisión de redes en torno a las máquinas virtuales, lo que, en suma, permite la configuración redes en la capa 2 sobre la capa 3, resultando en una mejora del rendimiento del tráfico de redes [24].

VxLAN extiende las capacidades de las VLAN, las cuales enfrentan problemas de escalabilidad. Mientras que las VLAN admiten un número aproximado de 4096 identificadores, VxLAN supera esta limitación al soportar hasta 16 000 000 de identificadores de túneles [25]. El encabezado del paquete de VxLAN está configurado para contener un campo de identificación

de segmento de 24 bits, motivo por el cual puede representar hasta 16 000 000 de segmentos virtuales diferentes.

La red virtual, al no ser visible para la red física, elimina la necesidad de tener una infraestructura física extra. Además, al considerar el mismo segmento de VxLAN, se reduce la duplicación de direcciones MAC para las máquinas virtuales existentes. De esta manera, las direcciones MAC pueden superponerse cuando no se encuentran en el mismo segmento VxLAN. En una red VxLAN, el enlace de datos que es parte del mismo segmento de VxLAN, tiene una dirección MAC única.

La configuración de VxLAN requiere el conocimiento de ciertos términos, como VTEP, que significa Punto Final Virtual de Túnel. Los VTEP corresponden a los gateways que se encargan de la encapsulación y desencapsulación de los paquetes. Estos VTEP pueden implementarse a modo de puente dentro de hipervisores, máquinas virtuales o switches que permitan la configuración de VxLAN [26]. Otro término importante es el de VNI o Identificador de Red VxLAN, que como su nombre lo indica, sirve para identificar cada segmento de red. El tráfico de un segmento VxLAN se aisla usando el VNI [27]. La estructura de un paquete VxLAN encapsulado se muestra en la Fig. 3.

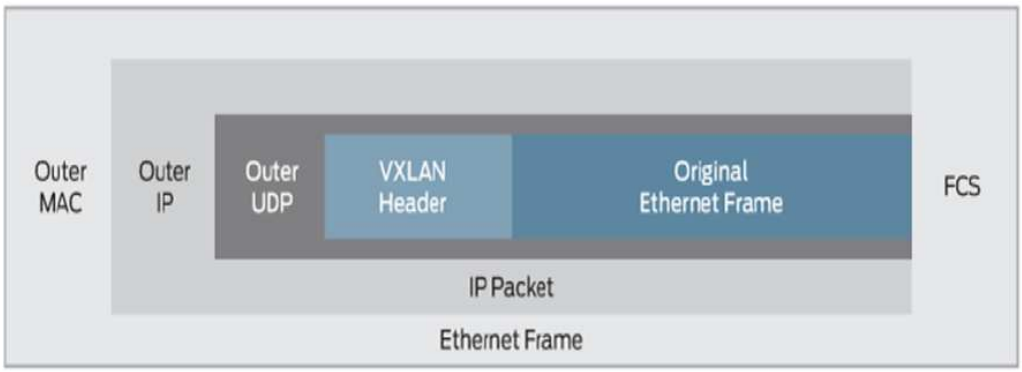


Fig. 3. Formato de paquete VxLAN encapsulado [28].

Según [26], los VTEP siguen tres reglas de reenvío. En primer lugar, no se realiza la encapsulación o desencapsulación del paquete de VxLAN siempre que dos máquinas virtuales se encuentren en el mismo host, y por tanto, la conmutación sea local. Luego, en el caso de que las máquinas se ubiquen en diferentes hosts, el tráfico se encapsula o desencapsula con el encabezado VxLAN en el VTEP de origen, y es el VTEP de destino que se encarga de desencapsular el paquete VxLAN, el cual a su vez lo envía a la máquina de recepción. Por último, para el tráfico

con origen desconocido, difusión o multidifusión, el VTEP añade un encabezado al paquete y lo envía como una trama encapsulada a la dirección asociada al VNI. En el VTEP de destino, el paquete es recibido y es tratado como tráfico unicast convencional.

Algunas de las desventajas y desafíos asociados con la implementación de VxLAN giran en torno a la necesidad de crear tablas extensas de mapeo en cada switch virtual. Esto se debe a la posibilidad de que la dirección MAC del servidor virtual de destino se encuentre en cualquier subred IP [29]. Además, los esquemas basados en VxLAN tienen la posibilidad de necesitar switches virtuales sofisticados, lo cual podría generar sobrecostos al crear planos de control utilizando controladores SDN, especialmente si estos son controladores propietarios.

C. Multiprotocol Label Switching

MPLS es una tecnología de túneles introducida por el Grupo de Trabajo de Ingeniería de Internet (IETF) que provee un mecanismo para configurar las Redes Privadas Virtuales (VPNs). El objetivo inicial de MPLS fue optimizar el direccionamiento de paquetes en redes de alto rendimiento, teniendo en cuenta que esta tecnología en lugar de reenviar los paquetes IP a los dispositivos finales lo hace a los enrutadores de destino acorde a pequeñas etiquetas [30].

Entre los tipos más conocidos de VPN se encuentran las VPN de Capa 2 y las de Capa 3. En las primeras, las tramas son enviadas de manera directa desde el origen al destino, mientras que en las segundas, la red se compone de un grupo de nodos interconectados a través de una red centralizada dada por MPLS. Con esta tecnología de túneles, al agregar un nuevo sitio a una red, solo se necesita actualizar el enrutador de borde del proveedor (PE) que ofrece servicios al borde del cliente (CE). Para admitir nuevos clientes, se utiliza una tabla de la tecnología Virtual Routing and Forwarding (VRF), la cual contiene información sobre las rutas de diferentes sitios [30].

La información de enrutamiento en L3VPN se distribuye a los PE utilizando protocolos de enrutamiento como el Protocolo de Puerta de Enlace de Borde (BGP). Por ejemplo, en el RFC 4364 [31], se intercambia la información de enrutamiento privado entre PEs mediante BGP interno (I-BGP), separada de otras VPN debido al atributo de Route Distinguisher [32].

D. Métricas de rendimiento

La evaluación del rendimiento entre los nodos de redes requiere del uso de métricas variadas, que en conjunto, evalúan el tiempo de transferencia de datos. Según [33], algunas de las métricas de rendimiento son: Latencia: Muestra el tiempo que un paquete tarda en transmitirse en una red. Su unidad de medida son los milisegundos. Throughput: Métrica que cuantifica los datos transferidos en una red desde un punto a otro en un segundo. Su unidad de medida es bit por segundo bps.

E. Metodología PPDIOO

De acuerdo con [34], PPDIOO es una metodología de CISCO que establece el ciclo de vida del servicio de una red. Las fases comprenden: Preparación, Planificación, Diseño, Implementación, Operación y Optimización. Este ciclo de vida posibilita que la gestión de la red se realice de forma organizada y especialmente efectiva en el cumplimiento de sus objetivos.

- Preparación: Se identifican las tecnologías que respaldarán la arquitectura.
- Planificación: Se identifican los requerimientos de red, así como el lugar y los posibles usuarios de la misma.
- Diseño: Se diseña la red con base en la información recopilada en las fases anteriores cumpliendo con los requerimientos técnicos y de negocio. El diseño incluye la lista de dispositivos, servicios de red y forma la base de las tareas de implementación.
- Implementación: Se construye la red con las especificaciones provistas en la etapa de Diseño, tratando que los dispositivos se integren correctamente sin ningún tipo de vulnerabilidad.
- Operación: Se administra la red y se monitorea el rendimiento de esta. Se analizan los cambios y la existencia de errores.
- Optimización: En esta fase se realizan las mejoras necesarias para que la red funcione tal como se había planeado. Los cambios pueden rediseñar la red en el caso que existan muchos errores o el rendimiento no sea el adecuado.

III. METODOLOGÍA/TÉCNICAS/DISEÑO

A. *Metodología de Desarrollo*

Como ya se mencionó, la metodología elegida es PPDIOO. Su utilidad recae en los entornos empresariales, en los cuales se necesita de una estricta planificación y un mantenimiento constante para tratar de asegurar la eficiencia de una red. El presente trabajo busca desarrollar cada una de las fases de la metodología.

1. Preparación

Esta etapa consiste en la identificación de las tecnologías requeridas para diseñar la red, así como en el análisis de sus características más importantes. Por ende, los siguientes componentes corresponden a los equipos y tecnologías que fueron utilizados:

GNS3: Este software emulador de redes se encuentra disponible tanto en una versión de escritorio para el Sistema Operativo Windows como en una máquina virtual de Ubuntu que actúa como servidor para alojar dispositivos o plantillas de GNS3. El servidor se configuró con 5 GB de memoria RAM.

Cisco IOS: Imagen de Router de la serie 7200 VXR NPE-400 de Cisco que se caracteriza por ofrecer un rendimiento notable en relación con su precio, además de brindar modularidad y escalabilidad. Esta serie es ideal para la agregación de servicios en el borde de una red WAN/MAN ya sea empresas o proveedores de servicios. Algunos de los aspectos más destacados del router incluyen el soporte de Cisco Express Fordwarding, QoS, MPLS (MPLS VPN, MPLS traffic engineering, MPLS QoS), Tunneling (L2TP, L2TPV3, ACL, entre otros), así como protocolos de Capa 2 y Capa 3 (TCP: Protocolo de Control de Transmisión, UDP: Protocolo de Datagramas de Usuario, ARP: Protocolo de Resolución de Direcciones), Protocolos de Enrutamiento de Capa 3 (OSPF, BGP, entre otros) y presenta un rendimiento de hasta los 400 kpps para el procesador NPE-400. En términos de memoria, cuenta con 512 MB de RAM y 128 MB de NVRAM, entre otras características [35].

Open vSwitch: Más conocido como OVS, Open vSwitch es un conmutador virtual de múltiples capas el cual se encuentra distribuido bajo la licencia de código abierto Apache 2.0. Entre los protocolos que admite se encuentran Netflow, sFlow, 802.1 ag, IPFIX, entre otros [36].

Contenedor Ubuntu: La imagen de Docker Ubuntu corresponde al sistema operativo Linux que se encuentra basado en Debian, en su versión 20. La descarga de la imagen puede ser accedida desde la tienda de GNS3 [37].

Switch: Conmutador que se encuentra incluido por defecto en el emulador de GNS3.

QEMU Firefox: Distribución ligera basada en Tyni Core Linux que contiene el navegador Firefox preinstalado.

OpenDaylight: Controlador SDN con distribución karaf 0.5.2 Boron SR2 obtenido de https://nexus.opendaylight.org/content/repositories/opendaylight.release/org/opendaylight/integration/distribution-karaf/0.5.2-Boron-SR2/distribution-karaf-0.5.2-Boron-SR2.zip. La versión de Boron fue lanzada en el año 2016 como resultado del trabajo conjunto entre usuarios y proveedores de equipos de red. Esta versión se optimizó para brindar soporte en la nube, así como para la ingeniería de redes complejas. Cabe mencionar que para que funcione OpenDaylight se necesita tener instalado el entorno de ejecución de Java más conocido como Java Runtime Environment que en este caso corresponde a la versión 8 [38].

2. Planificación

Conforme a las tecnologías detalladas en la etapa de preparación, se llevó a cabo el plan de diseño para SDN. Dado que se trata de una emulación de red, los posibles usuarios son los hosts virtuales. En relación al número de usuarios, en este caso se crea un túnel VxLAN con dos nodos extremos, no obstante, como ya se mencionó con la tecnología VxLAN es posible generar 16 000 000 de identificadores de redes virtuales, cada uno de los cuales ofrece un espacio de segmentación independiente por lo que el número de posibles beneficiarios es alto. El plan contiene las tareas a realizar, como se muestra en la Tabla II.

En el primer mes se llevó a cabo la preparación y planificación; en el segundo mes se realizaron las fases de diseño e implementación de red; el tercer mes se dedicó a la etapa de operación y, finalmente, en el cuarto mes, se operó la red emulada con las modificaciones requeridas.

TABLA II

PLAN DE TRABAJO PARA EL DISEÑO DE UNA RED DEFINIDA POR SOFTWARE CON TECNOLOGÍA VXLAN PARA LA OPTIMIZACIÓN DE REDES EMPRESARIALES.

Fase	**Mes 1**	**Mes 2**	**Mes 3**	**Mes 4**
Preparación	X			
Planificación	X			
Diseño de red		X		
Implementación		X		
Operación			X	
Optimización				X

3. Diseño:

Con base en la información de las etapas anteriores, se procedió a diseñar la topología de red, la cual se compone de una red Underlay (subyacente) MPLS conformada por dos routers P y dos routers PE, así como dos routers CE ubicados en diferentes sitios. Cada uno de ellos cuales cuenta con un contenedor Open vSwitch, un controlador OpenDaylight, un switch y una PC (Ubuntu Docker Guest). La red Overlay (superpuesta) VxLAN se configura entre los switches virtuales Open vSwitch, como se puede observar en la Fig. 4.

Con base en dicha topología, se establece el direccionamiento IP para cada uno de los dispositivos, como se detalla en la Tabla III.

TABLA III

DIRECCIONAMIENTO IP DE LOS DISPOSITIVOS.

Dispositivo	**Interfaz**	**Dirección IP**	**Máscara de subred**	**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	

Continúa en la siguiente página...

Tabla III – continuación de la página anterior

Dispositivo	Interfaz	Dirección IP	Máscara de subred	Gateway
	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 vSwitch1	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 vSwitch2	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	
OpenDaylight1	Eth0	192.168.100.2	255.255.255.0	192.168.100.1
OpenDaylight2	Eth0	192.168.200.2	255.255.255.0	192.168.200.1
PC1 (Ubuntu Docker Guest1)	E0	192.168.1.2	255.255.255.0	192.168.1.1
PC2 ((Ubuntu Docker Guest2))	E0	192.168.1.3	255.255.255.0	192.168.1.1

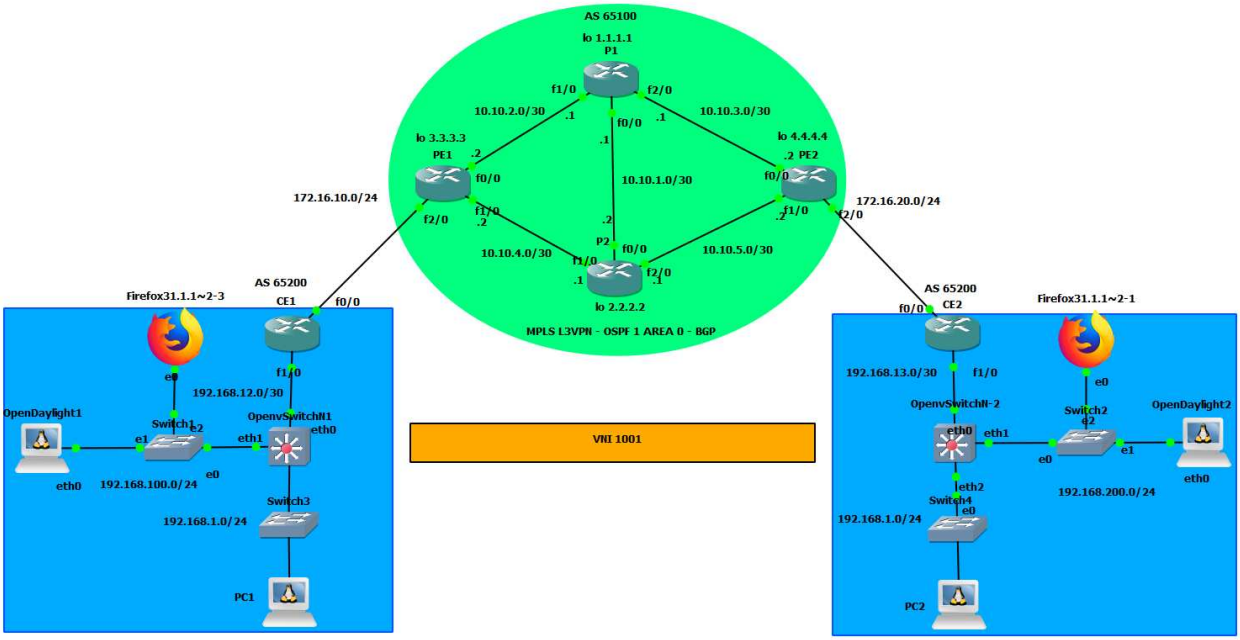


Fig. 4. Topología de red de prueba

4. *Implementación:*

La construcción de la red siguió las consideraciones de la etapa de diseño. El primer paso consitió en la configuración de las direcciones IP para el router P1, que se encuentra en el núcleo de la red Underlay. Para establecer las direcciones IP se accedió a cada interfaz mediante el comando “interface [nombre de la interfaz]”. Luego, se asignó una dirección IP con el comando “ip address [IP] [máscara]”. Posteriormente, se activó la interfaz con el comando “no shutdown” y se guardaron las configuraciones con “do write”. Para salir del modo configuración, se utilizó el comando “exit”. Para mostrar los resultados de las configuraciones de las direcciones IP, se utilizó el comando “show ip interface brief”, como se observa en la Fig. 5.

```
P1#sh ip interface brief
Interface              IP-Address      OK? Method Status                Protocol
FastEthernet0/0        10.10.1.1       YES NVRAM  up                    up
FastEthernet1/0        10.10.2.1       YES NVRAM  up                    up
FastEthernet2/0        10.10.3.1       YES NVRAM  up                    up
FastEthernet3/0        unassigned      YES NVRAM  administratively down down
Loopback0              1.1.1.1         YES manual up                    up
```

Fig. 5. Asignación de direcciones IP en el router P1

Después de configurar las interfaces, se estableció el enrutamiento dinámico interno utilizando el protocolo OSPF. Se utilizó el comando “router ospf 1” para establecer OSPF con

el ID 1 en el router, el comando “router-id 1.1.1.1” para asignar un identificador, el comando “network 1.1.1.1 0.0.0.0 area 0” para definir a la red 1.1.1.1 como parte del área 0, “network 10.10.0.0 0.0.255.255 area 0” para definir la red 10.10.0.0 como parte del área 0. Se debe tener en cuenta que este proceso se repite en los demás routers que son parte de la red Underlay. Para mostrar los resultados de enrutamiento, se utiliza el comando “sh ip route” como se observa en la Fig. 6.

```
P1#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
C        1.1.1.1 is directly connected, Loopback0
      2.0.0.0/32 is subnetted, 1 subnets
O        2.2.2.2 [110/2] via 10.10.1.2, 00:12:21, FastEthernet0/0
      3.0.0.0/32 is subnetted, 1 subnets
O        3.3.3.3 [110/2] via 10.10.2.2, 00:21:45, FastEthernet1/0
      4.0.0.0/32 is subnetted, 1 subnets
O        4.4.4.4 [110/3] via 10.10.1.2, 00:10:51, FastEthernet0/0
      10.0.0.0/8 is variably subnetted, 8 subnets, 2 masks
C        10.10.1.0/30 is directly connected, FastEthernet0/0
L        10.10.1.1/32 is directly connected, FastEthernet0/0
C        10.10.2.0/30 is directly connected, FastEthernet1/0
L        10.10.2.1/32 is directly connected, FastEthernet1/0
C        10.10.3.0/30 is directly connected, FastEthernet2/0
L        10.10.3.1/32 is directly connected, FastEthernet2/0
O        10.10.4.0/30 [110/2] via 10.10.2.2, 00:21:45, FastEthernet1/0
                      [110/2] via 10.10.1.2, 00:11:17, FastEthernet0/0
O        10.10.5.0/30 [110/2] via 10.10.1.2, 00:10:51, FastEthernet0/0
```

Fig. 6. Enrutamiento del router P1

El siguiente paso realizado fue habilitar MPLS, para lo cual fue necesario emplear el comando “ip cef”. La configuración del protocolo de distribución de etiquetas, o LDP, se llevó a cabo en cada de una de las interfaces mediante el comando “mpls ip”. Otra opción es configurarlo dentro del router ospf 1 utilizando el comando “mpls ldp autoconfig” de esta forma, LDP se configura automáticamente. Para visualizar los resultados de la configuración de mpls, se utiliza el comando “sh mpls interfaces” como se muestra en la Fig. 7.

También se especificó el rango de etiquetas que el router puede asignar y recibir. En este caso, se definió el rango de 100 a 199, para lo cual se utilizó el comando “mpls range 100 199”. Para verficar esta configuración, se empleó el comando “sh run — include mpls”, como se puede observar en la Fig. 8.

```
P1#sh mpls interfaces
Interface              IP           Tunnel   BGP Static Operational
FastEthernet0/0        Yes (ldp)    No       No  No     Yes
FastEthernet1/0        Yes (ldp)    No       No  No     Yes
FastEthernet2/0        Yes (ldp)    No       No  No     Yes
```

Fig. 7. Configuración de MPLS en P1

```
P1#sh run | include mpls
mpls label range 100 199
 mpls ip
 mpls ip
 mpls ldp autoconfig
mpls ldp router-id Loopback0
```

Fig. 8. Rango de etiquetas mpls en P1

El proceso de configuración del router P2 es similar al router P1. En la Fig. 9 se observa la configuración de las direcciones IP de las interfaces del router.

```
P2#sh ip interface brief
Interface              IP-Address      OK? Method Status                Protocol
FastEthernet0/0        10.10.1.2       YES NVRAM  up                    up
FastEthernet1/0        10.10.4.1       YES NVRAM  up                    up
FastEthernet2/0        10.10.5.1       YES NVRAM  up                    up
FastEthernet3/0        unassigned      YES NVRAM  administratively down down
Loopback0              2.2.2.2         YES manual up                    up
```

Fig. 9. Asignación de direcciones IP en el router P2

Los resultados de enrutamiento del router P2 se muestra en la Fig. 10.Además, en este router se configuró el rango de etiquetas de 420 a 450 como se muestra en la Fig. 11.

Después de la configuración de los routers P1, continuó la configuración de los routers PE. En el router PE1, se asignaron las direcciones IP en cada interfaz, como se observa en la Fig. 12, siguiendo las direcciones propuestas en la etapa de Diseño.

El siguiente paso fue la implementación del VRF con el fin de crear una instancia virtual de la tabla de enrutamiento. En este caso, se creó una instancia VRF llamada CLIENTEA utilizando el comando “ip vrf CLIENTEA”. Luego, se asignó el Route Distinguisher a la VRF para garantizar la unicidad global con el comando “rd 65200:1”. Por otro lado, “route-target export 65200:1” es un comando que sirve para exportar las rutas de esta VRF, mientras que el comando “route-target import 65200:2” indica que las rutas con el Route Distinguisher:2 se importan en el VRF. Esta configuración se visualiza en la Fig. 13 mediante el comando “sh run vrf”.

```
P2#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 10.10.1.1, 00:13:52, FastEthernet0/0
      2.0.0.0/32 is subnetted, 1 subnets
C        2.2.2.2 is directly connected, Loopback0
      3.0.0.0/32 is subnetted, 1 subnets
O        3.3.3.3 [110/2] via 10.10.4.2, 00:12:53, FastEthernet1/0
      4.0.0.0/32 is subnetted, 1 subnets
O        4.4.4.4 [110/2] via 10.10.5.2, 00:12:26, FastEthernet2/0
      10.0.0.0/8 is variably subnetted, 8 subnets, 2 masks
C        10.10.1.0/30 is directly connected, FastEthernet0/0
L        10.10.1.2/32 is directly connected, FastEthernet0/0
O        10.10.2.0/30 [110/2] via 10.10.4.2, 00:12:53, FastEthernet1/0
                      [110/2] via 10.10.1.1, 00:13:52, FastEthernet0/0
O        10.10.3.0/30 [110/2] via 10.10.5.2, 00:12:26, FastEthernet2/0
C        10.10.4.0/30 is directly connected, FastEthernet1/0
L        10.10.4.1/32 is directly connected, FastEthernet1/0
C        10.10.5.0/30 is directly connected, FastEthernet2/0
L        10.10.5.1/32 is directly connected, FastEthernet2/0
```

Fig. 10. Enrutamiento del router P2

```
P2#sh run | include mpls
mpls label range 420 450
 mpls ldp autoconfig
mpls ldp router-id Loopback0
```

Fig. 11. Rango de etiquetas mpls en router P2

```
PE1#sh ip interface brief
Interface              IP-Address      OK? Method Status                Protocol
FastEthernet0/0        10.10.2.2       YES NVRAM  up                    up
FastEthernet1/0        10.10.4.2       YES NVRAM  up                    up
FastEthernet2/0        172.16.10.1     YES NVRAM  up                    up
FastEthernet3/0        unassigned      YES NVRAM  administratively down down
Loopback0              3.3.3.3         YES NVRAM  up                    up
```

Fig. 12. Asignación de direcciones IP en el router PE1.

```
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
!
!
```

Fig. 13. Configuración de VRF CLIENTEA en el router PE1

Otro punto esencial es la configuración del protocolo de enrutamiento interno OSPF en los routers PE, con el que se establecen las redes a las que se conecta el router, en este caso, dentro de la red MPLS. La configuración de OSPF es importante para el posterior intercambio de etiquetas MPLS. El resultado de dicha configuración se muestra en la Fig. 14.

```
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
```

Fig. 14. Configuración del protocolo OSPF en el router PE1

En este caso, también se necesita la configuración del protocolo BGP para la conexión con redes externas. Se estableció la configuración de este protocolo con el comando “router bgp 65100”, utilizando ASN 65100. Además, se determinó la identificación del router mediante del comando “bgp router-id 4.4.4.4”. Posteriormente, se habilitó el registro de cambios en la vecindad BGP a través del comando “bgp log-neighbor-changes”.

Se configuró un vecino BGP correspondiente al router PE2, para ello se utilizó el comando neighbor “3.3.3.3 remote-as 65100”. De la misma manera, se definió a la interfaz Loopback 0 como la interfaz de origen para las actualizaciones BGP hacia el vecino antes definido utilizando el comando “neighbor 3.3.3.3 update-source Loopback 0”. Se utilizó el comando “exit” para continuar con la configuración de la familia de direcciones VPNv4. Esta primera parte de

configuración del protocolo BGP en el router PE1 se visualiza en la Fig. 15.

```
router bgp 65100
 bgp router-id 3.3.3.3
 bgp log-neighbor-changes
 neighbor 4.4.4.4 remote-as 65100
 neighbor 4.4.4.4 update-source Loopback0
 !
```

Fig. 15. Configuración inicial del protocolo BGP en el router PE1

Para la configuración de la familia de direcciones VPNv4, se empleó el comando “address-family vpnv4”. De la misma forma, se activó la vecindad de BGP y se configuró el envío de atributos de comunidad extendidos al router vecino con los comandos “neighbor 3.3.3.3 activate” y “neighbor 3.3.3.3 send-community extended”, respectivamente. La configuración de la familia de direcciones VPNv4, se finalizó con el comando “exit-address-family”.

Posteriormente, fue necesaria la configuración del vrf CLIENTEA. Para ello, se empleó el comando “address-family ipv4 vrf CLIENTEA”, en el mismo se configuró un vecino BGP con la dirección IP 172.16.20.2 y un ASN remoto 65200 perteneciente al router CE1 mediante el comando “neighbor 172.16.10.2 remote-as 65200”. La vecindad se activó con el comando “neighbor 172.16.10.2 activate”. La configuración finalizó con el comando “exit-address-family”. Los resultados finales de la configuración de BGP en PE1 se muestran en la Fig. 16.

```
router bgp 65100
 bgp router-id 3.3.3.3
 bgp log-neighbor-changes
 neighbor 4.4.4.4 remote-as 65100
 neighbor 4.4.4.4 update-source Loopback0
 !
 address-family vpnv4
  neighbor 4.4.4.4 activate
  neighbor 4.4.4.4 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf CLIENTEA
  neighbor 172.16.10.2 remote-as 65200
  neighbor 172.16.10.2 activate
 exit-address-family
!
```

Fig. 16. Configuración final del protocolo BGP en router PE1

Al igual que en los router P, en el router PE1 también se configuró la tecnología MPLS, resultado que se muestra en la Fig. 17.

La configuración del router PE2 fue similar a la realizada en el router PE1. De esta

```
PE1#sh run | include mpls
 mpls ldp autoconfig
mpls ldp router-id Loopback0
```

Fig. 17. Configuración de MPLS en el router PE1

manera se inició con la asignación de las direcciones IP a las interfaces como se muestra en la Fig. 18.

```
PE2#sh ip interface brief
Interface              IP-Address      OK? Method Status                Protocol
FastEthernet0/0        10.10.3.2       YES NVRAM  up                    up
FastEthernet1/0        10.10.5.2       YES NVRAM  up                    up
FastEthernet2/0        172.16.20.1     YES NVRAM  up                    up
FastEthernet3/0        unassigned      YES NVRAM  administratively down down
Loopback0              4.4.4.4         YES NVRAM  up                    up
```

Fig. 18. Asignación de direcciones IP en el router PE2

Se siguió con la configuración del VRF del mismo CLIENTEA, pero en este caso en el Route Distinguisher se asignó 65200:2 para asegurar la unicidad global. Se exportaron las rutas del rd 65200:2, mientras que se importaron las del rd 65200:1, como se observa en la Fig. 19.

```
PE2#sh run vrf
Building configuration...

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

Fig. 19. Configuración de VRF CLIENTEA en PE2

También se configuró el protocolo de enrutamiento OSPF cuya tabla de enrutamiento final se muestra en la Fig. 20.

De la misma manera, se configuró el protocolo BGP invirtiendo el router vecino con respecto al router PE1. Dicha configuración se observa en la Figura 21.

Finalmente, en este router se configuró la tecnología MPLS de la misma forma que se lo hizo en el router PE1. El resultado se muestra en la Fig. 22.

Luego, se configuraron los routers CE. Al igual que los routers anteriores, en el router CE1 la configuración inició con la asignación de las direcciones IP en las interfaces FastEthernet y Loopback que se aprecian en la Fig. 23.

```
PE2#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 10.10.3.1, 00:33:45, FastEthernet0/0
      2.0.0.0/32 is subnetted, 1 subnets
O        2.2.2.2 [110/2] via 10.10.5.1, 00:22:43, FastEthernet1/0
      3.0.0.0/32 is subnetted, 1 subnets
O        3.3.3.3 [110/3] via 10.10.5.1, 00:22:43, FastEthernet1/0
                 [110/3] via 10.10.3.1, 00:33:45, FastEthernet0/0
      4.0.0.0/32 is subnetted, 1 subnets
C        4.4.4.4 is directly connected, Loopback0
      10.0.0.0/8 is variably subnetted, 7 subnets, 2 masks
O        10.10.1.0/30 [110/2] via 10.10.5.1, 00:22:43, FastEthernet1/0
                      [110/2] via 10.10.3.1, 00:25:59, FastEthernet0/0
O        10.10.2.0/30 [110/2] via 10.10.3.1, 00:33:45, FastEthernet0/0
C        10.10.3.0/30 is directly connected, FastEthernet0/0
L        10.10.3.2/32 is directly connected, FastEthernet0/0
O        10.10.4.0/30 [110/2] via 10.10.5.1, 00:22:43, FastEthernet1/0
C        10.10.5.0/30 is directly connected, FastEthernet1/0
L        10.10.5.2/32 is directly connected, FastEthernet1/0
```

Fig. 20. Configuración del protocolo OSPF en el router PE2

```
router bgp 65100
 bgp router-id 4.4.4.4
 bgp log-neighbor-changes
 neighbor 3.3.3.3 remote-as 65100
 neighbor 3.3.3.3 update-source Loopback0
 !
 address-family vpnv4
  neighbor 3.3.3.3 activate
  neighbor 3.3.3.3 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf CLIENTEA
  neighbor 172.16.20.2 remote-as 65200
  neighbor 172.16.20.2 activate
 exit-address-family
!
```

Fig. 21. Configuración final del protocolo BGP en router PE2

```
PE2#sh run | include mpls
 mpls ldp autoconfig
mpls ldp router-id Loopback0
```

Fig. 22. Configuración de MPLS en el router PE2

```
CE1#sh ip interface brief
Interface              IP-Address      OK? Method Status                Protocol
FastEthernet0/0        172.16.10.2     YES NVRAM  up                    up
FastEthernet1/0        192.168.12.1    YES NVRAM  up                    up
FastEthernet2/0        unassigned      YES NVRAM  up                    down
FastEthernet3/0        unassigned      YES NVRAM  administratively down down
Loopback0              172.16.1.10     YES NVRAM  up                    up
```

Fig. 23. Asignación de direcciones IP en el router CE1

A continuación, se configuró el protocolo de enrutamiento BGP con el número de sistema autónomo ASN 65200. Se habilitó la función de registro para los cambios respecto a los vecinos BGP con el comando “bgp log-neighbors-changes”. Se anunció la red correspondiente a la dirección de la Loopback por medio de BGP utilizando el comando “network 172.16.1.10 mask 255.255.255.255”. El uso de la máscara de subred 255.255.255.255 establece el anuncio de una sola dirección IP, que en este caso es 172.16.1.10. Con el comando anterior, también se anunció la red que se conecta al Docker de Open vSwitch.

Posteriormente, se indicó que el router PE1 con la dirección IP 172.16.10.1 es vecino BGP con el número ASN 65100 mediante el comando “neighbor 172.16.10.1 remote-as 65100”. También se aceptan las rutas que contienen el número ASN del router vecino mediante el comando “neighbor 172.16.10.1 allowas-in”. Estas configuraciones se visualizan en la Fig. 24.

```
router bgp 65200
 bgp log-neighbor-changes
 network 172.16.1.10 mask 255.255.255.255
 network 192.168.12.0 mask 255.255.255.252
 neighbor 172.16.10.1 remote-as 65100
 neighbor 172.16.10.1 allowas-in
```

Fig. 24. Configuración de BGP en router CE1

Las configuraciones del router CE1 se repiten en el router CE2. Se asignaron las direcciones IP de las interfaces como se observa en la Fig. 25.

```
CE2#sh ip interface brief
Interface              IP-Address      OK? Method Status                Protocol
FastEthernet0/0        172.16.20.2     YES NVRAM  up                    up
FastEthernet1/0        192.168.13.1    YES NVRAM  up                    up
FastEthernet2/0        unassigned      YES NVRAM  administratively down down
FastEthernet3/0        unassigned      YES NVRAM  administratively down down
Loopback0              172.16.2.20     YES NVRAM  up                    up
```

Fig. 25. Asignación de direcciones IP en el router CE2

De la misma forma se configuró el protocolo de enrutamiento BGP en el router CE2 con el ASN 65200 para conectar con el router PE2 que tiene ASN 65100. Las configuraciones se

muestran en la Fig. 26.

```
router bgp 65200
 bgp log-neighbor-changes
 network 172.16.2.20 mask 255.255.255.255
 network 192.168.13.0 mask 255.255.255.252
 neighbor 172.16.20.1 remote-as 65100
 neighbor 172.16.20.1 allowas-in
```

Fig. 26. Configuración de BGP en router CE2

Después, se procedió con la configuración de los contenedores dockers Open vSwitch. Se asignaron las direcciones IP en las tres interfaces que se conectan con el router CE, el switch que a su vez está conectado con el controlador SDN y la PC en una LAN. La asignación de las direcciones IP se puede realizar de forma directa mediante la opción “Edit Config” o mediante el uso del comando “ifconfig eth0 [dirección IP] netmask [máscara de red]”. Cabe mencionar que el resultado del enrutamiento en este y en los demás contenedores Ubuntu se lo puede realizar a través del comando “ip route”. En la Fig. 27 se visualizan las direcciones IP en el Open vSwitchN1, para lo que se utilizó el comando “ifconfig [nombre de interfaz]”.

```
/ # ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 72:A7:88:73:85:7E
          inet addr:192.168.12.2  Bcast:0.0.0.0  Mask:255.255.255.252
          inet6 addr: fe80::70a7:88ff:fe73:857e/64 Scope:Link
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:9666 errors:0 dropped:1 overruns:0 frame:0
          TX packets:42 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:996249 (972.8 KiB)  TX bytes:3081 (3.0 KiB)

/ # ifconfig eth1
eth1      Link encap:Ethernet  HWaddr BA:9C:DC:AD:4E:C8
          inet addr:192.168.100.1  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::b89c:dcff:fead:4ec8/64 Scope:Link
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:16183 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:1149574 (1.0 MiB)

/ # ifconfig eth2
eth2      Link encap:Ethernet  HWaddr FE:B4:F5:CC:D0:05
          inet addr:192.168.1.1  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::fcb4:f5ff:fecc:d005/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7814 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:406766 (397.2 KiB)
```

Fig. 27. Asignación de direcciones IP en el OpenvSwitchN1

Para la configuración de la tecnología VxLAN, en primer lugar, se creó un puente, bridge o switch virtual con el nombre “ovs-br0” para lo que se utilizó el comando “ovs-vsct add-br ovs-br0”. Luego, se añadió un puerto llamado “vxlan0” al puente “ovs-br0” con el comando “ovs-vsctl add-port ovs-br0 vxlan0 – set interface vxlan0 type=vxlan options:remote_ip=192.168.13.2

options:key=1001". Este comando también define a la interfaz "vxlan0" como tipo vxlan con la dirección IP remota 192.168.13.2 y la clave o VNI 1001.

En caso de que el puente "br0", que se crea automáticamente, contenga a la interfaz eth2 (que se encuentra en la red LAN de la PC), se elimina este puerto de dicho puente y se lo añade al nuevo puente con los siguientes comandos: "ovs-vsctl del-port br0 eth2" y "ovs-vsctl add-port ovs-br0 eth2", respectivamente. Por último, se estableció el puerto "eth2" en la VLAN 10 con el comando "ovs-vsctl set port eth2 tag=10". Para visualizar estas configuraciones, se utiliza el comando ovs-vsctl show como se muestra en la Fig. 28.

Fig. 28. Configuración del puente ovs-br0 para establecer VxLAN en OpenvSwitchN1

En la Fig. 28, se visualiza que el puente "ovs-br0" está controlado por un dispositivo con dirección IP 192.168.100.2. Para ello, se utilizó el comando "ovs-vsctl set bridge ovs-br0 stp_enable=true", que habilita el Protocolo de Árbol de Expansión STP para evitar que se produzcan bucles en el puente ovs-br0. Posteriormente, se indicó que el puente debía utilizar la versión 1.3 del protocolo OpenFlow con el comando "ovs-vsctl set bridge ovs-br0 protocols=OpenFlow13". Finalmente, se estableció el controlador para el puente con conexión TCP, dirección IP 192.168.100.2 y puerto 6633 mediante el comando "ovs-vsctl set-controller ovs-br0 tcp:192.168.100.2:6633".

Las configuraciones del contenedor Docker OpenvSwitchN2 se establecieron de forma similar al OpenvSwitchN1. En la Fig. 29 se muestra las direcciones IP de las interfaces.

Las configuraciones del puente "ovs-br0" del contenedor Docker OpenvSwitchN2 se realizaron de modo similar al OpenvSwitchN1. Se consideró la dirección IP del dispositivo remoto con el que se crea el túnel VxLAN, que es es 192.168.12.2, y la dirección IP del controlador, que es 192.168.200.2, como se muestra en la Fig. 30.

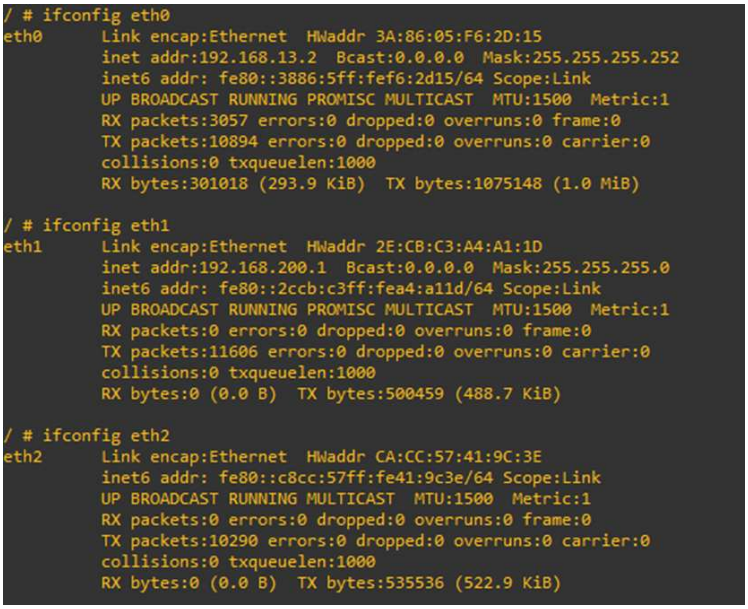


Fig. 29. Asignación de direcciones IP en el OpenvSwitchN2

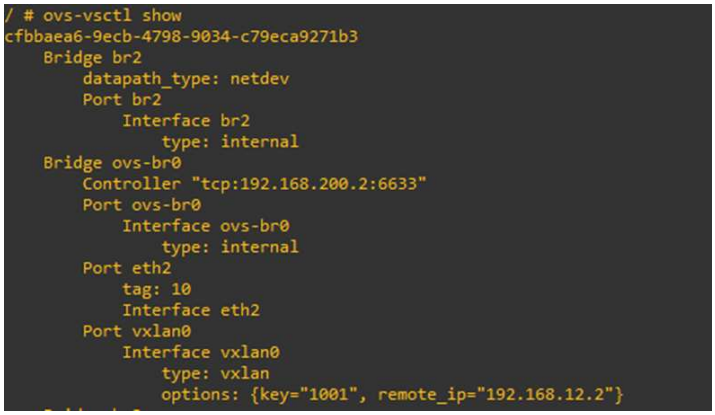


Fig. 30. Configuración del puente ovs-br0 para establecer VxLAN en OpenvSwitchN1

Luego, se configuró el controlador SDN OpenDaylight. En primer lugar, se utilizó el comando “cd distribution-karaf-0.5.2-Boron-SR2” para ubicarse en dicho directorio. Como se muestra en la Fig. 31 se exportaron las variables de entorno de “java_home” y “Path”. Con el comando “./bin/karaf” se ejecutó el programa. Estos comandos se replicaron en el otro controlador.

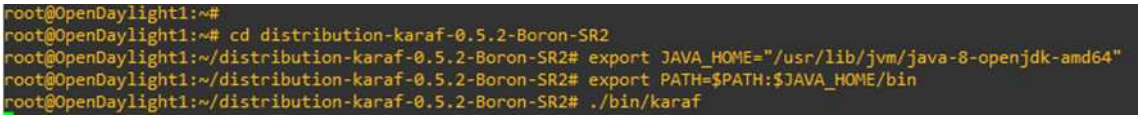


Fig. 31. Iniciación de OpenDaylight

Una vez dentro de la consola de OpenDaylight, se instalaron características adicionales. Entre ellas se encuentran “odl-restconf” que implementa Restconf, un protocolo basado en REST que permite el control de recursos YANG a través de HTTP. También se incluyeron “L2Switch”, que integra funciones de capa 2, “Mdsal-apidocs”, que proporciona documentación de API para el Modelo de Datos del Almacén MD-SAL, y “DLUX”, que brinda una interfaz de usuario web para gestionar la topología de red y otros aspectos del controlador. La línea de comandos utilizada se visualiza en la Fig. 32.

Fig. 32. Ejecución de OpenDaylight

Finalmente, se asignaron las direcciones IP a las PCs (Ubuntu Docker Guest) teniendo en cuenta que se encuentran en una red LAN Virtual extendida, por lo que la interfaz eth0 de la PC1 recibió la dirección IP 192.168.1.2 y la PC2 la dirección IP 192.168.1.3. Para la asignación de dichas direcciones, se utilizó el comando “ip addr add [dirección IP/máscara de red] dev eth0”. La Fig. 33 y la Fig. 34 muestran el resultado del uso de dichos comandos.

```
root@PC1:~# ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.2  netmask 255.255.255.0  broadcast 0.0.0.0
        inet6 fe80::1c19:bdff:fea1:494  prefixlen 64  scopeid 0x20<link>
        ether 1e:19:bd:a1:04:94  txqueuelen 1000  (Ethernet)
        RX packets 354  bytes 18738 (18.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 13  bytes 1006 (1.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Fig. 33. Interfaz eth0 de la PC1

```
root@PC2:~# ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.3  netmask 255.255.255.0  broadcast 0.0.0.0
        inet6 fe80::3089:a7ff:fe43:6e30  prefixlen 64  scopeid 0x20<link>
        ether 32:89:a7:43:6e:30  txqueuelen 1000  (Ethernet)
        RX packets 357  bytes 18894 (18.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 13  bytes 1006 (1.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Fig. 34. Interfaz eth0 de la PC2

5. Operación:

En esta fase, se monitorizó el estado de red. Se hizo ping de pc a pc para verificar la correcta implementación de las tecnologías antes mencionadas. En la Fig. 35, se muestra la verificación de la comunicación desde la PC1 a la 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
```

Fig. 35. Ping de PC1 a PC2

Para visualizar la gestión del plano de control de la red en el controlador OpenDaylight, se abrió la imagen de QEMU que contiene el navegador Firefox. Luego, se seleccionó la opción del panel de control, “network” y dentro del mismo se establecieron los parámetros que se observan en la Fig. 36 con el fin de poder conectarse al controlador.

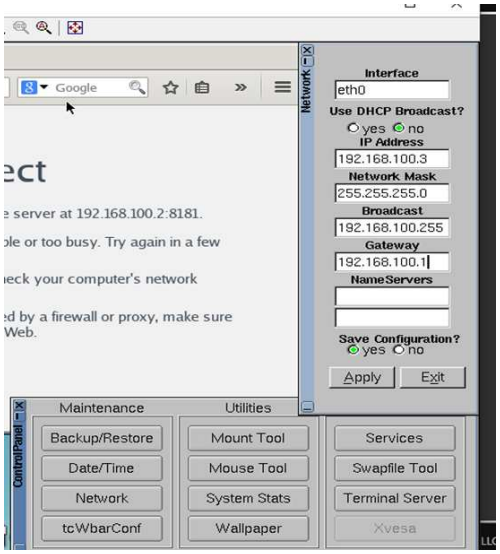


Fig. 36. Definición de parámetros de conexión en imagen QEMU Firefox

Una vez definidos los parámetros de conexión, en el navegador se estableció la dirección IP del controlador y el puerto 8181, seguido de la página de inicio de sesión, como se observa en la Fig. 37. Se debe tener en cuenta que las credenciales por defecto son admin:admin.

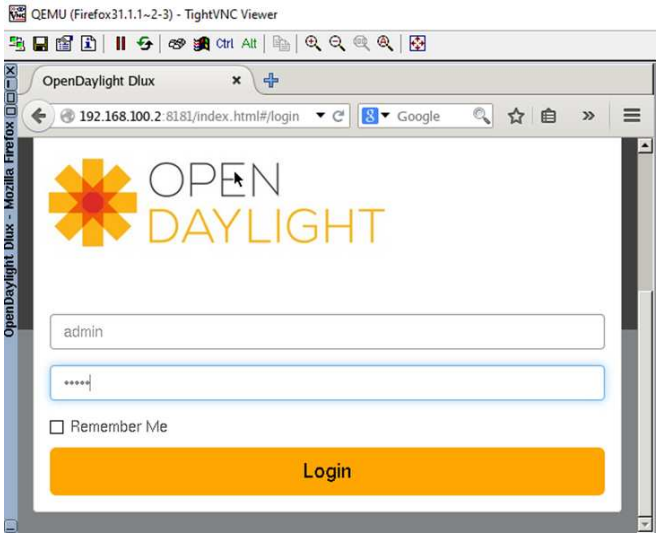


Fig. 37. Inicio de sesión en OpenDaylight

Al ingresar a la Interfaz de Usuario Web de OpenDaylight, se observó la existencia del dispositivo (Open vSwitch) con el protocolo OpenFlow 1.3. La Fig. 38 muestra dicha interfaz con las funcionalidades generales del controlador.

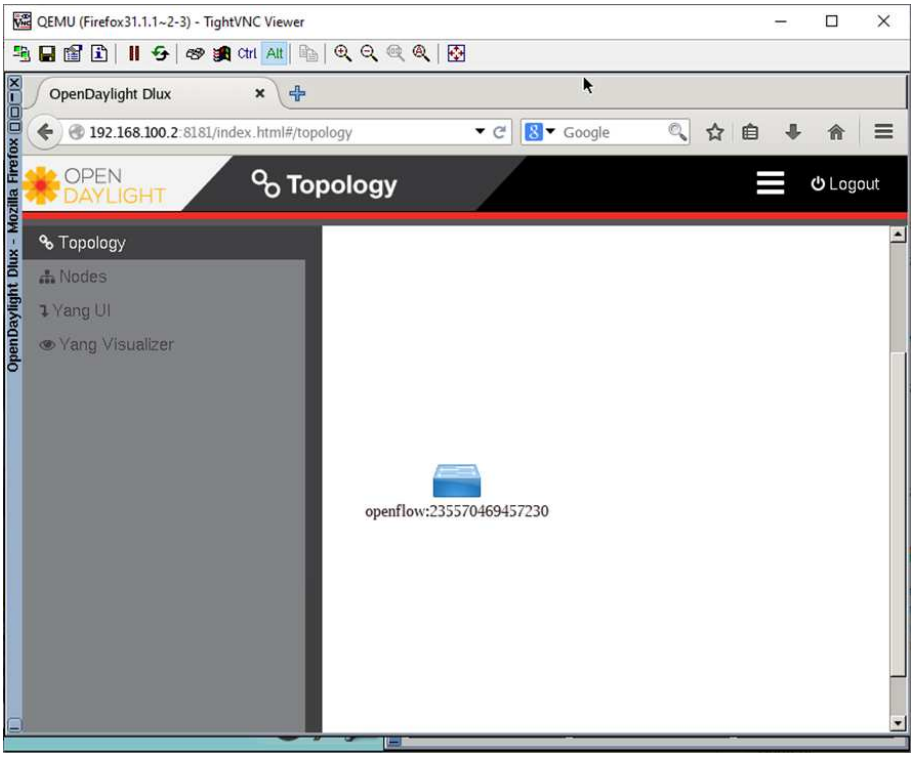


Fig. 38. Visualización de dispositivo con el protocolo OpenFlow

6. Optimización:

Una de las acciones realizadas para optimizar la red está relacionada con la seguridad de los routers. Se estableció una contraseña segura para el modo privilegio. De la misma forma, se configuró una contraseña en la línea de consola y se habilitó la autenticación. Asimismo, se configuró una contraseña para el acceso mediante las líneas de VTY, estableciendo dicho acceso únicamente mediante el protocolo SSH, y de igual forma, se habilitó la autenticación. Otro mecanismo de seguridad fue la configuración del nombre de dominio del sistema y la generación de claves criptográficas RSA para servicios como SSH. En la Fig. 39, se visualizan los comandos utilizados para la implementación de seguridad en los routers Cisco de la topología.

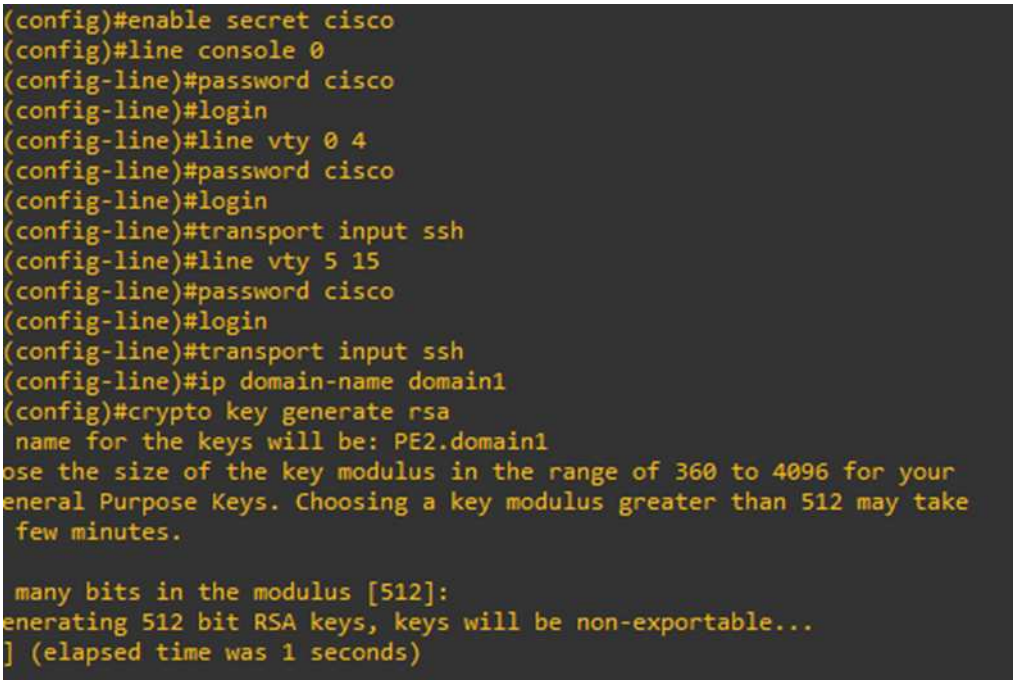


Fig. 39. Configuraciones de seguridad en routers

De igual forma, se agregaron reglas de flujo en los puentes ovs-br0 de ambos Open vSwitch para gestionar el tráfico de red. La primera regla establece la prioridad en 100 y se aplica al tráfico entrante por el puerto eth2, y se establece como acción la salida de dicho tráfico por el puerto vxlan0. La segunda regla se aplica al tráfico entrante por el puerto vxlan0, y se indica como salida el puerto eth2. Por último, se establece el puente ovs-br0 en modo seguro, con el fin de que esté funcionando siempre que el controlador esté activo. Estas optimizaciones se observan en la Fig. 40.

```
/ # 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
```

Fig. 40. Optimizaciones de reglas de flujo en los switches virtuales Open vSwitch

Otra acción de optimización realizada fue el incremento de la Unidad de Transmisión Máxima (MTU) en las interfaces de todos los dispositivos por donde fluye el tráfico de red. Se estableció el mtu en 1530 debido a que es el valor máximo permitido en los routers c7200. La optimización se evidencia en la Fig. 41 y Fig. 42.

```
P1(config)#interface fastEthernet 0/0
P1(config-if)#mtu 1530
P1(config-if)#exit
P1(config)#interface fastEthernet 1/0
P1(config-if)#mtu 1530
P1(config-if)#exit
P1(config)#interface fastEthernet 2/0
P1(config-if)#mtu 1530
P1(config-if)#do wr
```

Fig. 41. Optimización de mtu en router

```
/ # ifconfig eth0 mtu 1530
/ # ifconfig eth2 mtu 1530
/ # ifconfig eth0
eth0      Link encap:Ethernet  HWaddr B2:A4:AD:AB:C0:FB
          inet addr:192.168.12.2  Bcast:0.0.0.0  Mask:255.255.255.252
          inet6 addr: fe80::b0a4:adff:feab:c0fb/64 Scope:Link
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1530  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2534 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:117034 (114.2 KiB)

/ # ifconfig eth2
eth2      Link encap:Ethernet  HWaddr CA:D3:A3:98:30:46
          inet addr:192.168.1.1  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::c8d3:a3ff:fe98:3046/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1530  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1021 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:53440 (52.1 KiB)
```

Fig. 42. Optimización de mtu en switch virtual Open vSwitch

Por último, para optimizar el rendimiento que con base a los resultados preliminares observados en la etapa de operación se observó alta latencia. Se decidió migrar el diseño de la red emulada del servidor con 5 GB de memoria a un servidor de GNS3 con mejores prestaciones de Hardware. La Fig. 43 muestra las características de este servidor, de entre las cuales destaca

la memoria total de 50 GB y Swap o espacio de intercambio de memoria de 2 GB. También se observan los niveles de uso de CPU, memoria, Swap, y promedio de carga cuando se ejecuta el proyecto de emulación. Esta optimzación es fundamental para la mejora del tiempo que tarda en volver un paquete de datos.

Fig. 43. Características del servidor GNS3

IV. RESULTADOS

Una vez realizado cada uno de los pasos de la metodología PPDIOO, se evalúan las métricas de operación de los dispositivos configurados en el software de emulación. Para llevar a cabo el análisis de resultados, se necesitan de herramientas analizadoras de paquetes y protocolos de red, tales como: Wireshark, ping e iperf3.

A. Estadísticas de transmisión de paquetes

Se utilizó la herramienta Wireshark para capturar y analizar los paquetes enviados entre los hosts PC1 y PC2. La Fig. 44 muestra los paquetes capturados, dentro de los cuales se observa que el primer paquete corresponde al Protocolo De Control de Mensajes de Internet (ICMP,) entre los hosts antes mencionados. En los detalles del paquete se refleja el Protocolo de Datagrama de Usuario (UDP) ya que VxLAN encapsula las tramas de la capa 2 en paquetes UDP.

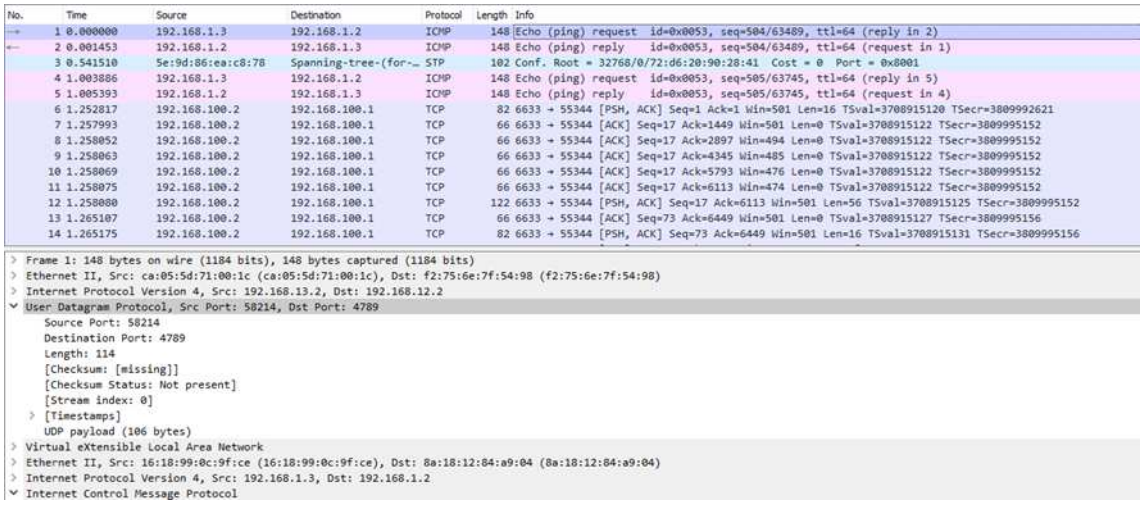


Fig. 44. Captura de paquetes en Wireshark

El paquete que se muestra en la Fig. 45 utiliza el Protocolo de Descubrimiento de Capa de Enlace (LLDP) y a nivel de detalle se verifica la información de la trama, así como el nombre del sistema que corresponde a openFlow e idenficador. También se observa que emplea VxLAN para la comunicación, así como el protocolo UDP.

Luego, en la barra de opciones de Wireshark se elige la opción Estadísticas, Gráficas de E/S y se filtra por VxLAN. De esta manera, en la Fig. 46 se observa un gráfico de barras que

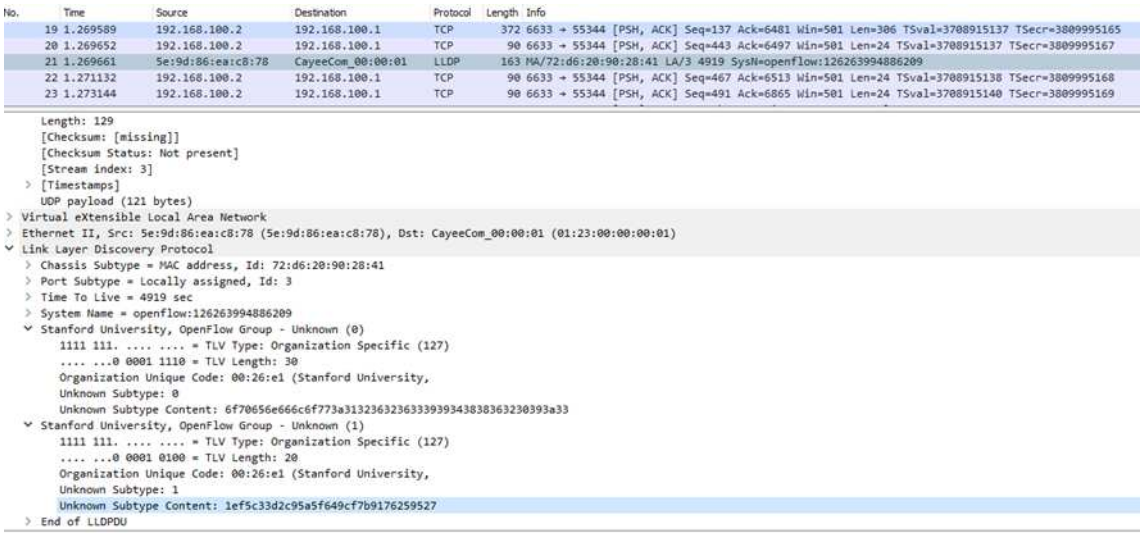


Fig. 45. Detalle de paquete capturado en Wireshark

proporciona información sobre el número de paquetes por segundo capturados en función del tiempo. El número más alto de paquetes capturados en un instante dado es 8.

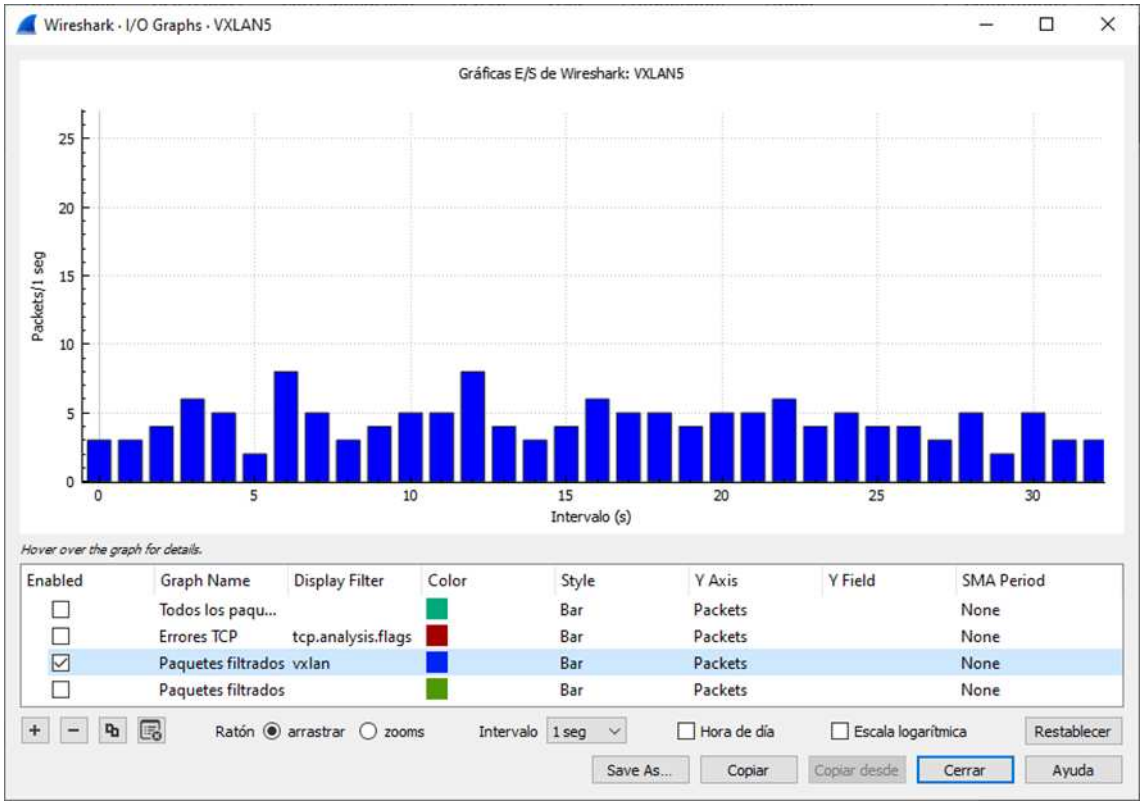


Fig. 46. Gráfico E/S de paquetes VxLAN

Otra información importante que provee Wireshark son las estadísticas de jerarquía de protocolo. Como se observa en la Fig. 47, se proporciona el porcentaje de paquetes, el número de paquetes totales, porcentaje de bytes, número de bytes, bits por segundo, paquetes de fin, bytes de fin y bits por segundo de fin. En cuanto a VxLAN, contempla el 98% de los paquetes UDP, con un total de 145 paquetes capturados durante el tiempo de evaluación, que equivalen a 1168 bytes con una tasa de transferencia de 287 bits por segundo.

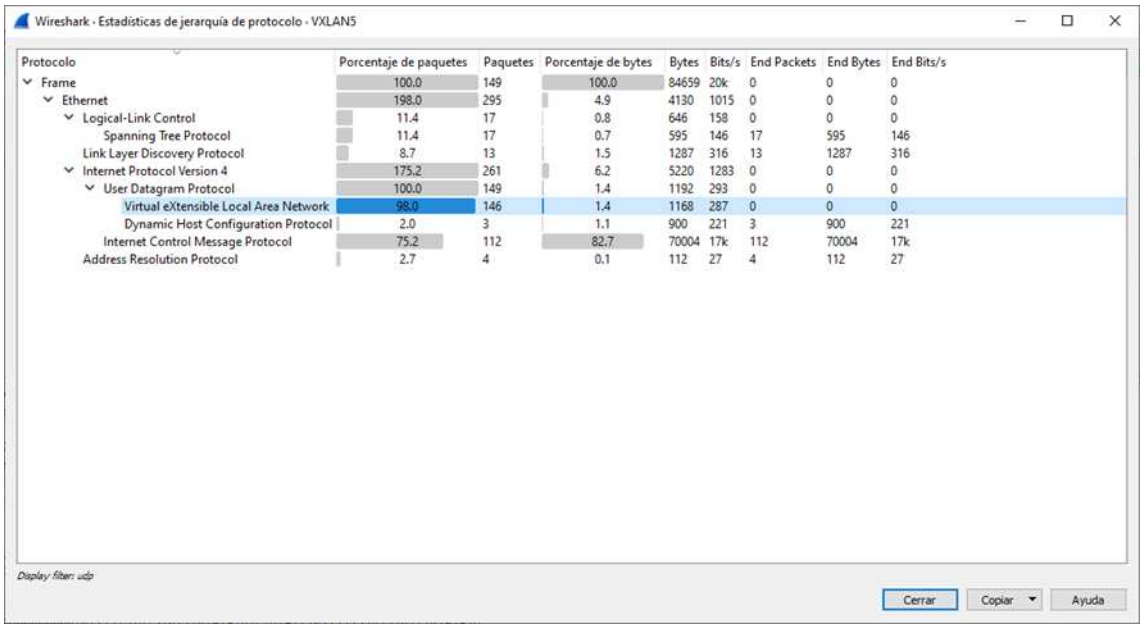


Fig. 47. Estadísticas de Jerarquía de Protocolo

También es importante analizar las estadísticas de los nodos finales de comunicación. En la Fig. 48 se muestran las estadísticas con el protocolo IPV4. Las PC o hosts finales tienen cada uno 112 paquetes (56 enviados y 56 recibidos) que equivalen a 79 000 bytes (aproximadamente 39 000 bytes enviados y 39 000 bytes recibidos).

En la Fig. 49, se muestran las estadísticas del Open vSwitchN1 y Open vSwitchN2 con el protocolo UDP. Se observa que por el puerto 4780, que corresponde a la interfaz vxlan0, se transfieren 64 y 82 paquetes, 40 000 y 42 000 bytes, respectivamente.

En cuanto al protocolo TCP, los nodos participantes corresponden esencialmente al controlador SDN y el Open vSwitch. Así, en la Fig. 50 se observa que en la conexión del controlador SDN dada por el puerto lógico 6633, se envían 234 paquetes equivalentes a 19 000 bytes, mientras que el Open vSwitch recibe dichos paquetes y bytes por el puerto lógico 55344.

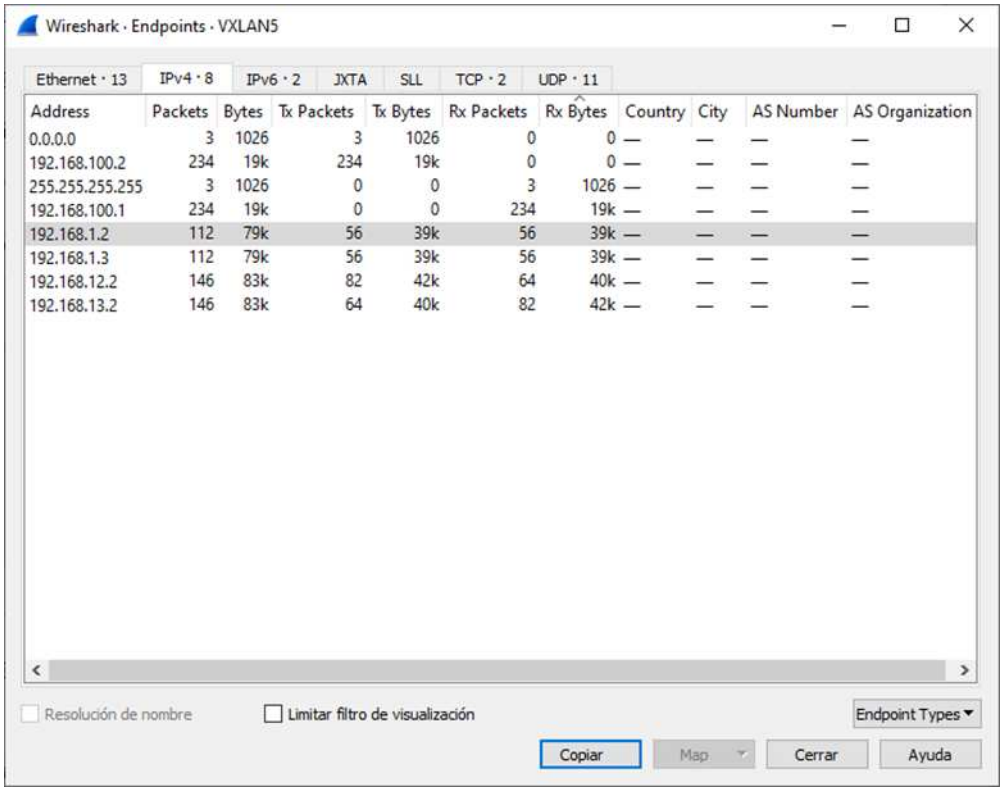


Fig. 48. Estadísticas de Nodos con el protocolo IPv4

B. Métricas de evaluación de rendimiento

Las métricas de evaluación de rendimiento que se consideraron fueron: Tiempo de Ida y Vuelta (RTT) para la latencia, Desviación estándar del RTT, Transferencia, Tasa de transferencia o Throughput, entre otras.

Utilizando la herramienta ping se realizaron tres pruebas de envío de paquetes desde la PC1 a la PC2, definiendo a la bandera (flag) -O, que establece la no fragmentación de paquetes. En caso de que un paquete sea muy grande para que no exista fragmentación, no se envían los paquetes. Se establece la dirección IP de destino 192.168.1.3, se indica el número de paquetes a enviar que en este caso fue de 50, con el flag -c 50, y mediante el flag -s 1422 se define el tamaño de los paquetes a enviar en 1422 bytes. La Fig. 51 indica los resultados obtenidos de la primera prueba. Los datos totales se representan en la Tabla IV.

Los resultados de la Tabla IV demuestran que la latencia promedio medida desde la PC1 a la PC2 oscila en un rango de 60,87 y 64,53 ms, y su desviación estándar se ubica entre los 4,68 y 13,36 ms.

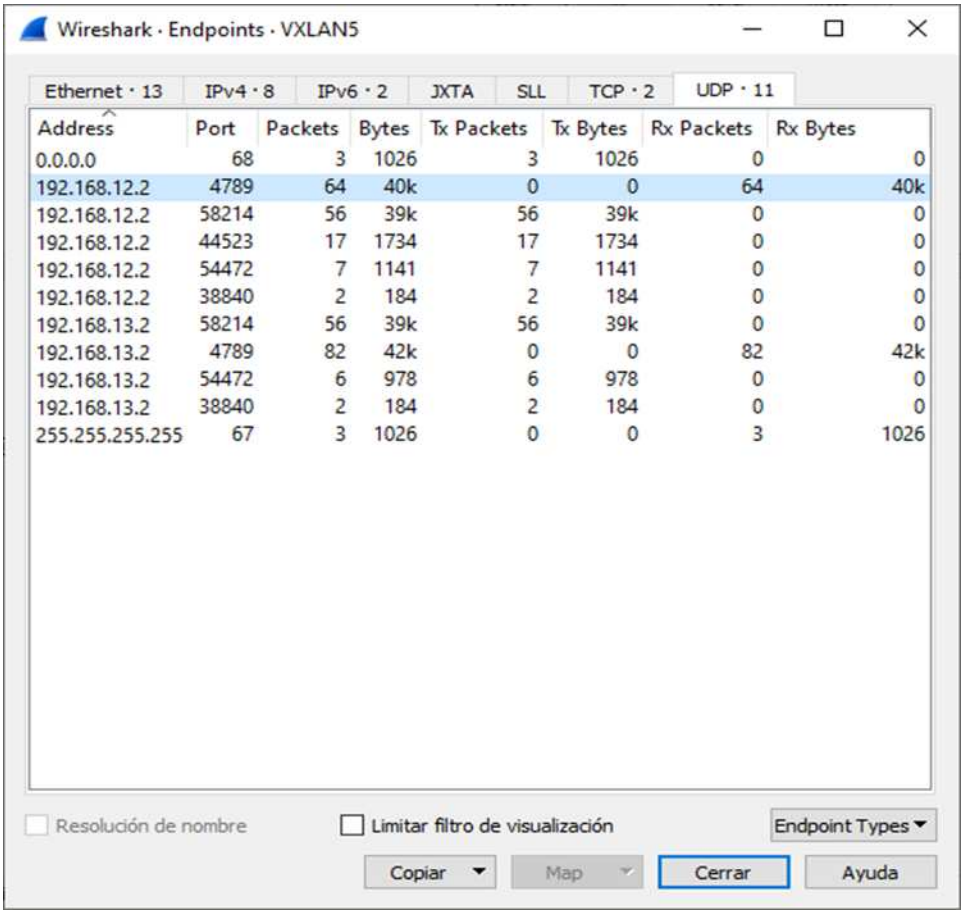


Fig. 49. Estadísticas de Nodos con el protocolo UDP

TABLA IV

RESULTADOS DE TRES PRUEBAS DE PING DESDE LA PC1 A LA PC2.

Prueba	RTT (ms)			
	Mínimo	Promedio	Máximo	Desviación estándar
1	43,27	64,53	137,60	13,36
2	51,82	61,23	83,38	7,26
3	52,87	60,87	73,3	4,68

Para la siguiente medición se utiliza la herramienta iperf3. Para ello, la PC2 se estableció como servidor mediante el comando iperf3 -s que trabaja en el puerto 5201, como se indica en la Fig. 52. Las pruebas se realizaron desde la PC1 a la PC2.

Desde la PC1 se utiliza el comando “iperf3 -c 192.168.1.3 -t 15 -i 5 -P 10 -u”, con el cual se indica que la PC1 actúa como cliente del servidor con dirección IP 192.168.1.3 (PC2), se indica el tiempo de duración de la prueba en 15 segundos, y se establece que se muestren los

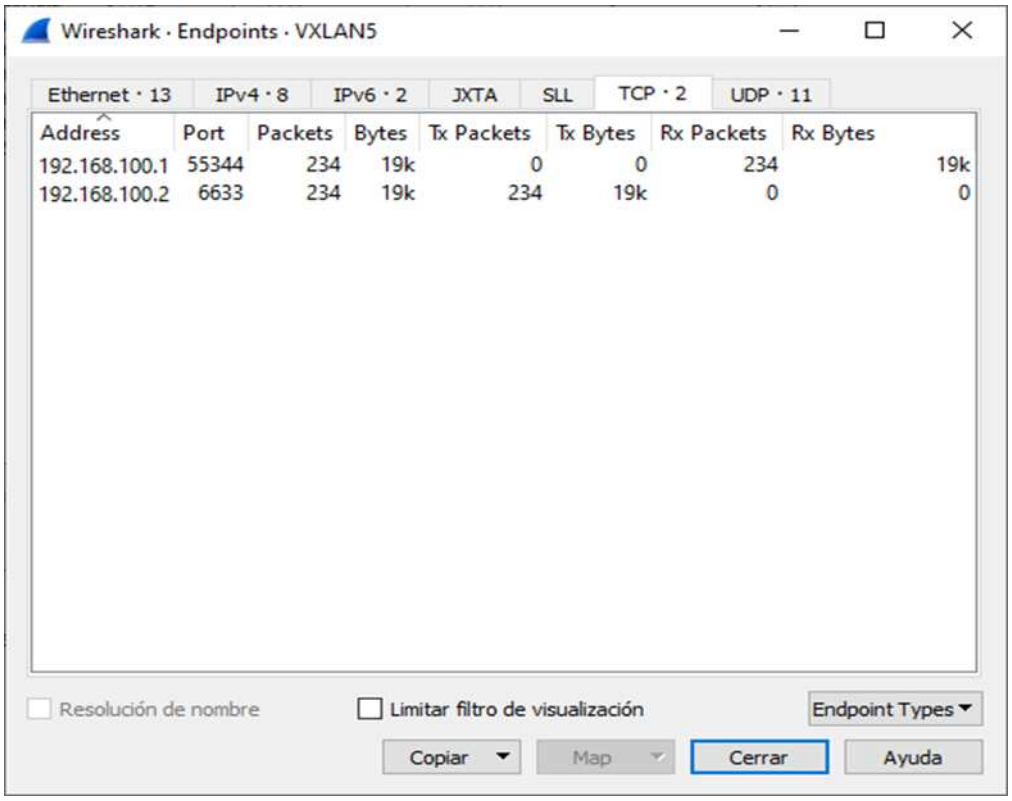


Fig. 50. Estadísticas de Nodos con el protocolo TCP

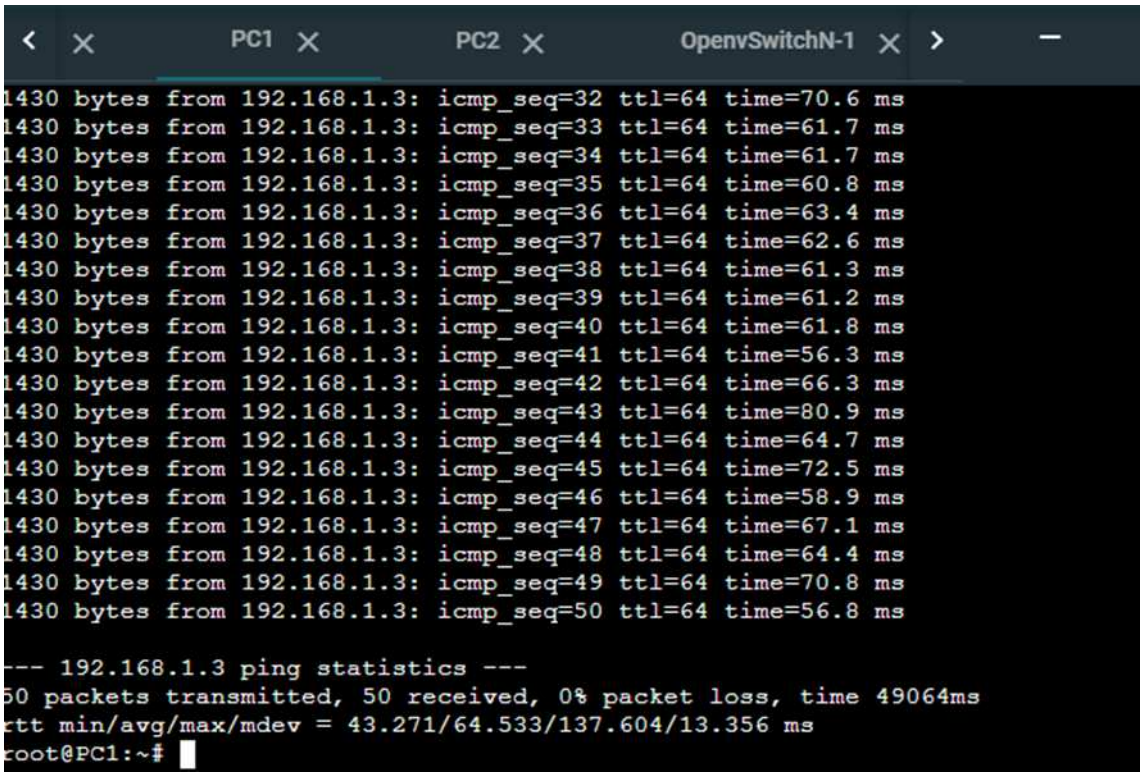


Fig. 51. Resultados de prueba de ping de PC1 a PC2

```
root@PC2:~# iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

Fig. 52. Servidor de herramienta IPERF3

resultados cada 5 segundos. Además, se indica 10 flujos paralelos para la prueba utilizando el protocolo UDP. Los resultados que se muestran en la Fig. 53, reflejan que en suma la cantidad total de datos transferidos bordean los 18,8 MB con una tasa de transferencia o throughput de 10,5 Mbits/sec.

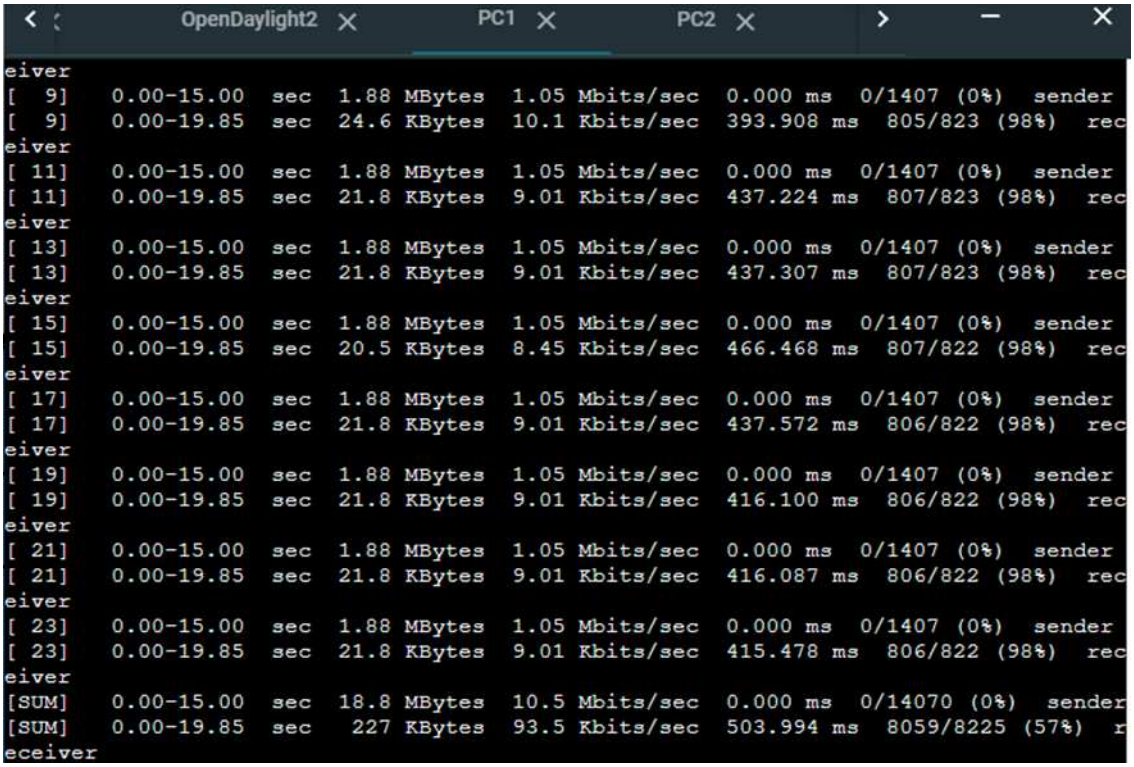


Fig. 53. Métricas de evaluación con el protocolo UDP

Con los parámetros anteriores se otra prueba pero esta vez el protocolo TCP. En la Fig. 54, se observa en las métricas que la cantidad de datos transferidos es 1,17 MB y la tasa de transferencia o throughput de 654 kbits/sec.

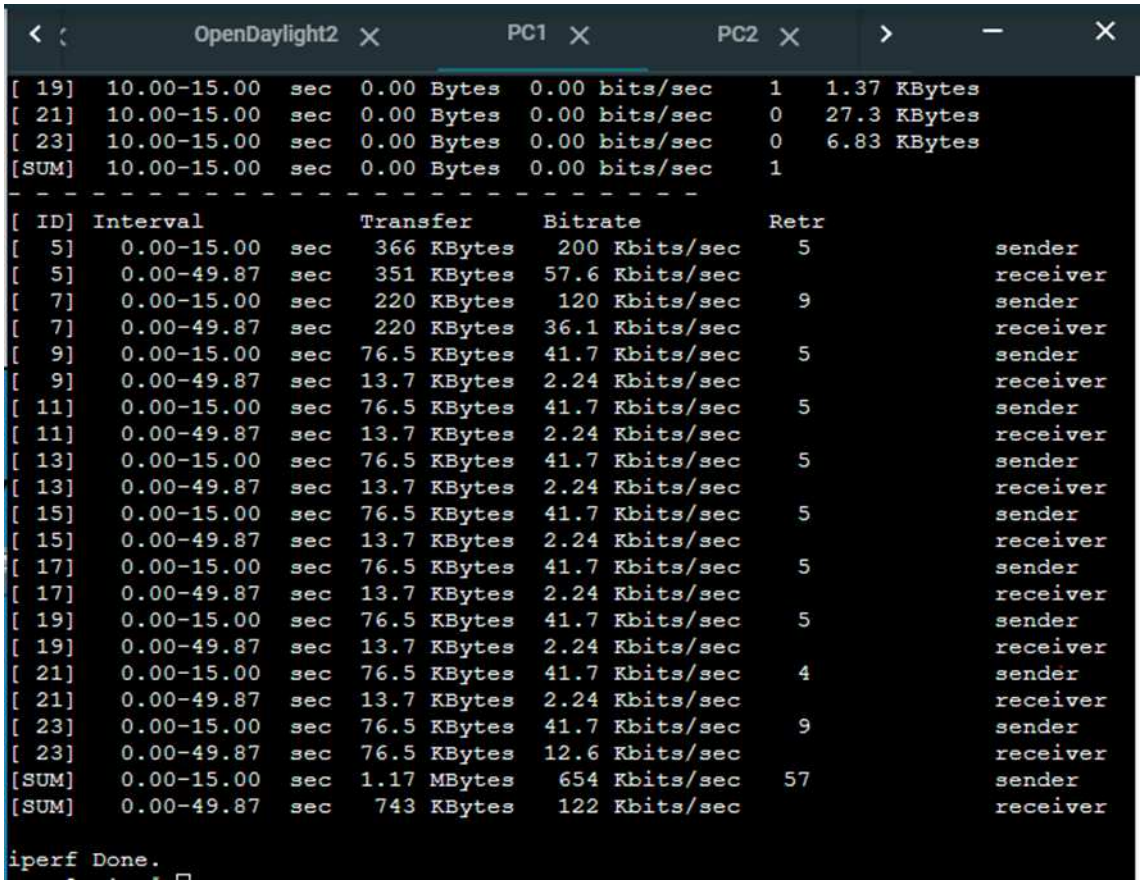


Fig. 54. Métricas de evaluación con el protocolo TCP

C. Seguridad y escalabiliad

En las redes configuradas con la tecnología VxLAN, generalmente solo pueden ser vulneradas desde adentro teniendo acceso a la LAN. Se conoce que las redes con VxLAN están superpuestas sobre infraestructura existente. Teniendo en cuenta que en la implementación se configuraron las VLAN. De esta manera, los VTEP envían el tráfico VxLAN a través de la VLAN, lo que establece un mecanismo de seguridad. La configuración de los bridges lógicos ovs-br0 en switches virtuales Open vSwitch con el modo secure permite que el tráfico se realice siempre que el controlador SDN se encuentre encendido. Además, dichos bridges segmentan el tráfico de red bajo reglas de flujos independientes.

Como se mencionó inicialmente, VxLAN permite tener 16 000 000 de túneles por lo que se puede configurar ese número de VNIs. Como se observa en la Fig. 55, en los detalles de los paquetes UDP capturados en Wireshark, se refleja el VNI 1001 que corresponde a un valor de

24 bits lo que permite configurar el número de túneles antes mencionados. Esto indica que la implementación de la tecnología VxLAN es altamente escalable.

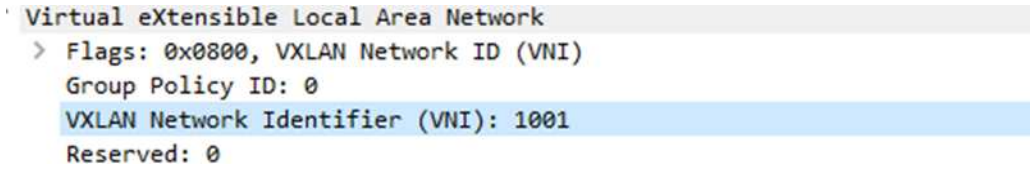


Fig. 55. Escalabilidad de VxLAN

El contar con un controlador SDN permite que la gestión de la red se realice de forma centralizada lo que hace que la red sea fácilmente escalable. Los dispositivos que implementan el protocolo OpenFlow se monitorizan mediante la visualización de estadísticas como ocurre en la Fig. 56. En ella se observa que el nodo (bridge lógico ovs-br0 del Open vSwitchN1) cuenta con tres puertos monitorizados. El puerto ovs-br0 que corresponde al puerto local. El puerto vxlan0 que es por donde se produce el tráfico encapsulado en VxLAN, y el puerto eth2, que es el puerto físico por el cual se conecta con la PC o host. Se observa la dirección MAC de cada puerto.

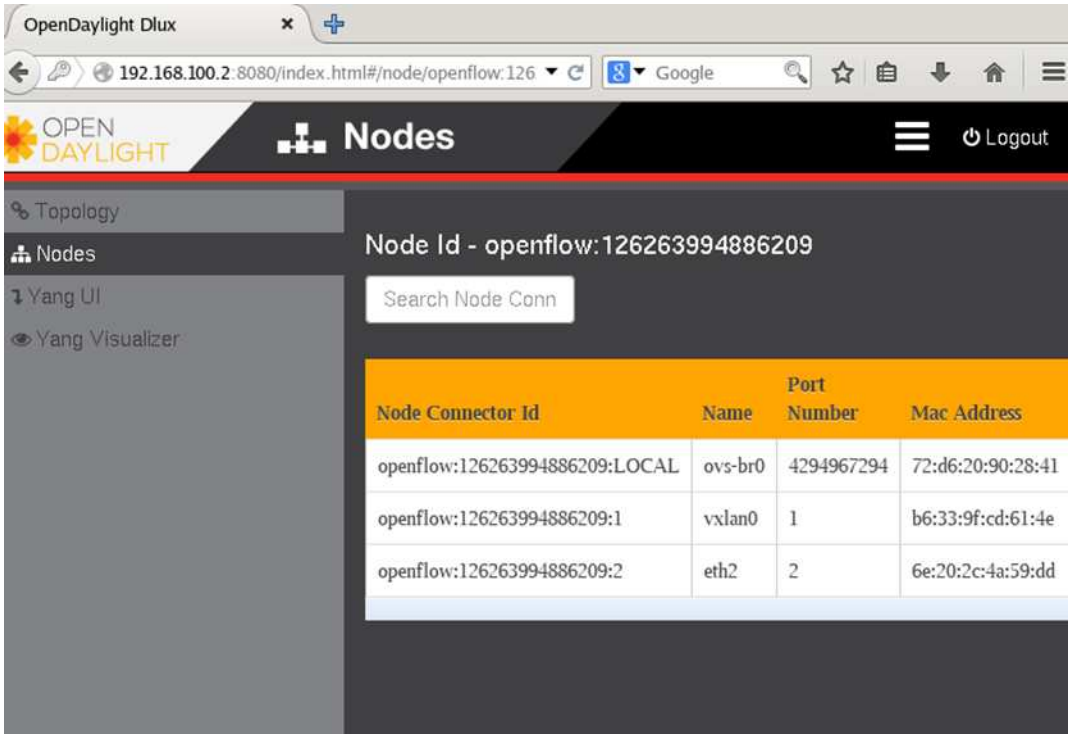


Fig. 56. Escalabilidad de una red SDN

V. Conclusiones y recomendaciones

A. *Conclusiones*

En este proyecto se logró diseñar una Red Definida por Software utilizando la tecnología VxLAN para la optimización de redes empresariales en un entorno emulado. La combinación de la tecnología VxLAN con SDN consolidan la seguridad y escalabilidad de una red, puesto que, por un lado, VxLAN es útil para segmentar una red y tener millones de usuarios que solo pueden comunicarse por el túnel con VNI asignado. Por otro, el controlador SDN es capaz de tener una visión global de la red, así como de gestionar las estadísticas de los bridges virtuales de dispositivos OpenFlow en los cuales se configura VxLAN. Estos bridges ofrecen la posibilidad de implementar reglas de flujo que controlan el paso de tráfico de red.

Se investigaron artículos relacionados al tema de principal del proyecto, y se destaca que la tecnología VxLAN en el contexto de las Redes Definidas por Software, tiene una gran utilidad por ser una tecnología virtualizada. No obstante, es necesario recalcar que existen pocas investigaciones al respecto ya que son tecnologías relativamente nuevas.

De acuerdo con las métricas obtenidas mediante las herramientas Ping e Iperf3, y considerando que el RTT es un indicador de la latencia de la red, se determina que la latencia promedio de la red diseñada se encuentra entre los 60,87 y 64,53 ms con una desviación estándar entre 4,68 y 13,36 ms, esto quiere decir que el tiempo de transmisión de paquetes es moderado para una red empresarial. Por su parte, la transferencia total datos en 15 segundos obtenida fue de 18,8 MB con una tasa de transferencia de comunicación o throughput que bordea los 10,5 Mbits/sec, que de igual forma se encuentra en los rangos aceptables. En la mayoría de las pruebas, el 100% de los paquetes enviados por un nodo fueron recibidos por el otro. Es necesario mencionar que los resultados están influenciados por las capacidades físicas del equipo sobre el cual se emuló la red, no obstante, las métricas demuestran eficiencia y rendimiento en la red diseñada.

Para llevar a cabo el proyecto se siguió la metodología PPDIOO, lo que fue clave para documentar y construir una topología de red que vaya acorde a los objetivos estipulados. Luego de optimizar la red, se analizaron las estadísticas de rendimiento mediante herramientas de análisis de tráfico de red como Wireshark, Ping e Iperf3. El uso de Wireshark fue fundamental para observar y verificar el funcionamiento de la red SDN en conjunto con la tecnología VxLAN, puesto que

en la captura de los paquetes se reflejaron los protocolos de comunicación encapsulados en la misma, tales como UDP por medio del cual se verificó la existencia del túnel VxLAN identificado con el VNI 1001, así como LLDP, que permitió comprobar la comunicación del switch virtual Open vSwitch con el controlador SDN a través del protocolo OpenFlow 1.3.

B. Recomendaciones

Las pruebas de evaluación de métricas se ven afectadas por el nivel de procesamiento de la máquina virtual sobre el cual se diseñó y emuló la red. Por tal motivo, se recomienda utilizar hardware con altos niveles de procesamiento (mayor RAM Y CPU) que mejore el rendimiento de la red.

Para la evaluación de métricas de rendimiento, se recomienda utilizar distintas herramientas de análisis de tráfico de red que permitan obtener una información amplia acerca de la eficiencia de una red. Es importante que se realicen varias pruebas de conexión y se capture un gran porcentaje de paquetes de datos para hacer una correcta medición.

Es importante conocer que el proceso de investigación para realizar una revisión de literatura debe llevarse a cabo en bases de datos bibliográficas de carácter científico. Es recomendable que estas bases estén relacionadas al área de redes de comunicación.

Es crucial que se realice cada una de las fases de la metodología PPDIOO para efectuar un proceso de diseño óptimo. Además, seguir las etapas de forma consecutiva facilita la documentación de la configuración de la red.

C. Trabajos futuros

Los routers c7200 poseen un mtu máximo de 1530 lo que limitó y condicionó las pruebas de envío de paquetes. Por tal razón, a futuro se puede probar este diseño de red utilizando routers que ofrezcan un mtu mayor.

El controlador SDN OpenDaylight fue útil para el diseño de la red. En trabajos futuros se puede considerar utilizar otros controladores o en su defecto, otras versiones del controlador empleado en este proyecto, para el uso de más funcionalidades que contribuyan a la simplificación de acciones de monitorización de dispositivos con el protocolo OpenFlow.

En este diseño se utilizaron plantillas de Ubuntu Docker Guest para instalar el controlador SDN, software Open vSwitch y simulación de hosts. A futuro, se puede considerar trabajar directamente con máquinas virtuales tal y como se trabaja en los entornos de virtualización que generalmente se utilizan en las redes empresariales.

Para tener mayor precisión en la evaluación de las métricas de rendimiento, se podría probar el diseño utilizando routers físicos que no estén supeditados a las capacidades de procesamiento de una máquina o servidor.

VI. Referencias

[1] Z. Zhao, F. Hong y R. Li, «SDN based VxLAN optimization in cloud computing networks,» *IEEE Access*, vol. 5, págs. 23 312-23 319, 2017.

[2] Z. Zhai, Q. Li et al., «Application Discussion of SDN Technology in Multi-data Center,» *Smart Systems and Green Energy*, vol. 2, nº 1, págs. 11-19, 2020.

[3] O. Blial, M. B. Mamoun y R. Benaini, «An Overview on SDN Architectures with Multiple Controllers,» *Journal of Computer Networks and Communications*, vol. 2016, Article ID 9396525, 2016. DOI: 10.1155/2016/9396525.

[4] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky y S. Uhlig, «Software-defined networking: A comprehensive survey,» *Proceedings of the IEEE*, vol. 103, nº 1, págs. 14-76, 2015, Cited by: 3463; All Open Access, Green Open Access. DOI: 10.1109/JPROC.2014.2371999. dirección: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84919935425&doi=10.1109%2fJPROC.2014.2371999&partnerID=40&md5=26b571f8a284cc9c0f48c3fc0e39a059.

[5] M. Hussain, N. Shah, R. Amin, S. S. Alshamrani, A. Alotaibi y S. M. Raza, «Software-Defined Networking: Categories, Analysis, and Future Directions,» *Sensors*, vol. 22, nº 15, 2022, Cited by: 13; All Open Access, Gold Open Access, Green Open Access. DOI: 10.3390/s22155551. dirección: https://www.scopus.com/inward/record.uri?eid=2-s2.0-%2085137145233&doi=10.3390%2fs22155551&partner%20ID=40&md5=e395742aab7f9e927c5d46f4f5e7ce00.

[6] D. A. S. GEORGE y A. H. George, «A Brief Overview of VXLAN EVPN,» *Ijireeicein-ternational Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering*, vol. 9, nº 7, págs. 1-12, 2021.

[7] G. D. Salazar-Chacón y A. R. R. García, «Segment-Routing Analysis: Proof-of-Concept Emulation in IPv4 and IPv6 Service Provider Infrastructures,» en *2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, IEEE, 2021, págs. 1-7.

[8] N. T. Seechurn, A. Mungur, S. Armoogum y S. Pudaruth, «Issues and Challenges for Network Virtualisation,» *International Journal of Communication Networks and Information Security*, vol. 13, nº 2, págs. 206-214, 2021, Cited by: 2. DOI: 10.54039/ijcnis.v13i2.4990.

dirección: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85115227815&doi=10.54039%2fijcnis.v13i2.4990&partnerID=40&md5=07576d3b4bfea66d17f06f1f68048dc8.

[9] B. Pfaff, J. Pettit, T. Koponen et al., «The Design and Implementation of Open vSwitch,» en *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, Oakland, CA: USENIX Association, mayo de 2015, págs. 117-130, ISBN: 978-1-931971-218. dirección: https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/pfaff.

[10] P. Emmerich, D. Raumer, F. Wohlfart y G. Carle, «Performance characteristics of virtual switching,» en *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)*, IEEE, 2014, págs. 120-125.

[11] B. Kitchenham y P. Brereton, «A systematic review of systematic review process research in software engineering,» *Information and Software Technology*, vol. 55, nº 12, págs. 2049-2075, 2013, ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2013.07.010. dirección: https://www.sciencedirect.com/science/article/pii/S0950584913001560.

[12] G. Salazar-Chacón, «Hybrid Networking SDN and SD-WAN: Traditional Network Architectures and Software-Defined Networks Interoperability in digitization era,» *Journal of Computer Science and Technology*, vol. 22, nº 1, e07-e07, 2022.

[13] D. Nuñez-Agurto, W. Fuertes, L. Marrone, E. Benavides-Astudillo y M. Vásquez-Bermúdez, «Traffic Classification in Software-Defined Networking by Employing Deep Learning Techniques: A Systematic Literature Review,» en *Technologies and Innovation*, R. Valencia-García, M. Bucaram-Leverone, J. Del Cioppo-Morstadt, N. Vera-Lucio y P. H. Centanaro-Quiroz, eds., Cham: Springer Nature Switzerland, 2023, págs. 67-80, ISBN: 978-3-031-45682-4.

[14] M. ÖZDEM y M. ALKAN, «SDN based management platform for intranet services,» *Politeknik Dergisi*, vol. 24, nº 3, págs. 989-995, 2021.

[15] M. B. Jimenez, D. Fernandez, J. E. Rivadeneira, L. Bellido y A. Cardenas, «A survey of the main security issues and solutions for the SDN architecture,» *IEEE Access*, vol. 9, págs. 122 016-122 038, 2021.

[16] V. Thirupathi, C. Sandeep, N. Kumar y P. P. Kumar, «A comprehensive review on sdn architecture, applications and major benifits of SDN,» *International Journal of Advanced Science and Technology*, vol. 28, nº 20, págs. 607-614, 2019.

[17] D. Nunez-Agurto, W. Fuertes, L. Marrone y M. Macas, «Machine Learning-Based Traffic Classification in Software-Defined Networking: A Systematic Literature Review, Challenges, and Future Research Directions.,» *IAENG International Journal of Computer Science*, vol. 49, nº 4, 2022.

[18] R. Klöti, V. Kotronis y P. Smith, «OpenFlow: A security analysis,» en *2013 21st IEEE International Conference on Network Protocols (ICNP)*, IEEE, 2013, págs. 1-6.

[19] M. Fernandez, «Evaluating OpenFlow controller paradigms,» en *ICN 2013, The Twelfth International Conference on Networks*, 2013, págs. 151-157.

[20] N. McKeown, T. Anderson, H. Balakrishnan et al., «OpenFlow: enabling innovation in campus networks,» *ACM SIGCOMM computer communication review*, vol. 38, nº 2, págs. 69-74, 2008.

[21] L. Zhu, M. M. Karim, K. Sharif, F. Li, X. Du y M. Guizani, «SDN controllers: Benchmarking & performance evaluation,» *arXiv preprint arXiv:1902.04491*, 2019.

[22] S. Rowshanrad, V. Abdi y M. Keshtgari, «Performance evaluation of SDN controllers: Floodlight and OpenDaylight,» *IIUM Engineering Journal*, vol. 17, nº 2, págs. 47-57, 2016.

[23] V. S. Raju, «SDN Controllers Comparison,» en *Proceedings of Science Globe International Conference*, jun. de 2018, págs. 1-5.

[24] P. Raj y A. Raman, «Software-Defined Network (SDN) for Network Virtualization,» en *Software-Defined Cloud Centers: Operational and Management Technologies and Tools*. Cham: Springer International Publishing, 2018, págs. 65-89, ISBN: 978-3-319-78637-7. DOI: 10.1007/978-3-319-78637-7_4. dirección: https://doi.org/10.1007/978-3-319-78637-7_4.

[25] M. S. Mulya Sudrajat, D. Perdana y R. M. Negara, «Performance Analysis of VXLAN and NVGRE Tunneling Protocol on Virtual Network,» *Bulletin of Electrical Engineering and Informatics*, vol. 6, nº 3, págs. 295-300, sep. de 2017.

[26] G. P. Reyes, M. Dammers y M. Kastanja, «Security assessment on a VXLAN-based network,» *Haettu*, vol. 10, nº 2017, págs. 2013-2014, 2014.

[27] M. Mahalingam, D. Dutt, K. Duda et al., *Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks*, RFC

7348, ago. de 2014. DOI: 10.17487/RFC7348. dirección: https://www.rfc-editor.org/info/rfc7348.

[28] T. Singh, V. Jain y G. S. Babu, «VXLAN and EVPN for data center network transformation,» en *2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, 2017, págs. 1-6. DOI: 10.1109/ICCCNT.2017.8203947.

[29] R. Chandramouli, «Analysis of Virtual Networking Options for Securing Virtual Machines,» *CLOUD COMPUTING 2016*, pág. 107, 2016.

[30] S. Mehraban, K. B. Vora y D. Upadhyay, «MPLS VPN using VRF (virtual routing and forwarding),» *International Journal of Advance Research, Ideas and Innovations in Technology*, vol. 3, nº 6, págs. 135-139, 2014.

[31] Y. Rekhter y E. C. Rosen, *BGP/MPLS IP Virtual Private Networks (VPNs)*, RFC 4364, feb. de 2006. DOI: 10.17487/RFC4364. dirección: https://www.rfc-editor.org/info/rfc4364.

[32] I. Martinez-Yelmo, D. Larrabeiti, I. Soto y P. Pacyna, «Multicast traffic aggregation in MPLS-based VPN networks,» *IEEE Communications Magazine*, vol. 45, nº 10, págs. 78-85, 2007.

[33] A. Shirmarz y A. Ghaffari, «Performance issues and solutions in SDN-based data center: a survey,» *The Journal of Supercomputing*, vol. 76, nº 10, págs. 7545-7593, 2020.

[34] P. Oppenheimer, *Top-Down Network Design*, 2010.

[35] Cisco, *Cisco 7200 VXR Series Routers Overview*, https://www.cisco.com/c/en/us/products/collateral/routers/7200-series-routers/data_sheet_c78_339749.pdf, Retrieved October 26, 2023, 2008.

[36] J. Grossmann, «Open vSwitch,» 2015, Retrieved October 26, 2023.

[37] J. Duponchelle, «Ubuntu Docker Guest,» 2017, Retrieved October 26, 2023.

[38] S. Badotra y J. Singh, «Open Daylight as a Controller for Software Defined Networking.,» *International Journal of Advanced Research in Computer Science*, vol. 8, nº 5, 2017.