Universidad Nacional De Loja

Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables

Carrera de Ingeniería en Telecomunicaciones

Desarrollo de una propuesta de diseño basada en redes definidas por software (SDN) para la red de datos de la Universidad Nacional Loja, campus la Argelia, ciudad de Loja

Trabajo de Integración curricular, previo a la obtención del título de Ingeniera en Telecomunicaciones.

AUTORA:

Diana Marisol Lozano Lozano

DIRECTOR:

Ing. Marco Suing Ochoa, Mg. Sc.

Loja – Ecuador

2024

Certificación

Loja, 17 de julio de 2024

Ing. Marco Suing Ochoa. M.Sc.
DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

CERTIFICO:

Que he revisado y orientado todo el proceso de elaboración del Trabajo de Integración Curricular denominado: **Desarrollo de una propuesta de diseño basada en redes definidas por software (SDN) para la red de datos de la Universidad Nacional Loja, campus la Argelia, ciudad de Loja**, previo a la obtención del título de **Ingeniero en Telecomunicaciones**, de la autoría de la estudiante **Diana Marisol Lozano Lozano**, con cédula de identidad **Nro.1950010247**, una vez que el trabajo cumple con todos los requisitos exigidos por la Universidad Nacional de Loja, para el efecto, autorizo la presentación del mismo para su respectiva sustentación y defensa.

Firmado electrónicamente por:
MARCO AUGUSTO SUING OCHOA

Ing. Marco Suing Ochoa. M.Sc.
DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Autoría

Yo, **Diana Marisol Lozano Lozano**, declaro ser autora del presente Trabajo de Integración Curricular y eximo expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos de posibles reclamos y acciones legales, por el contenido del mismo. Adicionalmente, acepto y autorizo a la Universidad Nacional de Loja la publicación de Trabajo de Integración Curricular en el Repositorio Digital Institucional-Biblioteca Virtual.

Firma:

Cédula:1950010247

Fecha: 17 de julio de 2024

Correo electrónico: diana.lozano@unl.edu.ec

Teléfono: 0988195080

Carta de autorización por parte de la autora, para consulta, reproducción parcial o total y/o publicación electrónica del texto completo del Trabajo de Integración Curricular.

Yo, **Diana Marisol Lozano**, declaro ser autora del Trabajo de Integración Curricular denominado: **Desarrollo de una propuesta de diseño basada en redes definidas por software (SDN) para la red de datos de la Universidad Nacional Loja, campus la Argelia, ciudad de Loja**, como requisito para optar por el título de **Ingeniera en Telecomunicaciones**, autorizo al sistema Bibliotecario de la Universidad Nacional de Loja para que, con fines académicos, muestre la producción intelectual de la Universidad, a través de la visibilidad de su contenido en el Repositorio Institucional.

Los usuarios pueden consultar el contenido de este trabajo en el Repositorio Institucional, en las redes de información del país y del exterior con las cuales tenga convenio la Universidad.

La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia del Trabajo de Integración Curricular que realice un tercero.

Para constancia de esta autorización, suscribo, en la ciudad de Loja, a los diecisiete días del mes de julio de dos mil veinticuatro.

Firma:

Autora: Diana Marisol Lozano Lozano

Cédula: 1950010247

Dirección: Loja

Correo electrónico: diana.lozano@unl.edu.ec

Teléfono: 0988195080

DATOS COMPLEMENTARIOS

Director del Trabajo de Integración Curricular: Ing. Marco Suing, Mg.Sc

Dedicatoria

Este trabajo de integración curricular representa el fruto de un viaje cargado de aprendizajes, diversas emociones y momentos de satisfacción. Durante mi trayectoria universitaria, ha sido fortalecida por un respaldo incondicional que merece reconocimiento.

Me permito dedicar a Dios por su constante guía y protección en cada paso de este recorrido académico. Dedicado a mi madre Elena Lozano, cuya confianza y fe inquebrantable en mi han sido un faro inspirador, a mis hermanos Manuel, José, Jorge, Sergio y Jhomaira Lozano, y familia en general, quienes han sido mi soporte incondicional a lo largo de mi formación profesional.

Los amigos constituyen un pilar fundamental en la existencia de cada individuo, y en la mía, han ocupado un espacio esencial. Por ello, dedico especialmente a Lizbeth M, Steffany R, Anabel C, Raquel P, Selena G y Erika P, cuya presencia constante y apoyo incondicional me han infundido ánimo y vitalidad en momentos cruciales.

La búsqueda constante de superación refleja mi firme determinación por alcanzar mis metas en esta travesía llamada vida. La perseverancia y la resiliencia han sido mis aliadas en la superación de los desafíos que se me han presentado en mi camino. Este logro es, es el resultado del sacrificio y del amor hacia mí.

Diana Marisol Lozano

Agradecimientos

La gratitud es un valor que he cultivado a lo largo de mi vida, y en este momento de celebración y logro, deseo expresar mi más sincero agradecimiento a quienes han contribuido en mi camino.

En primer lugar, elevo mi más profundo agradecimiento a Dios, cuya mano divina ha guiado cada paso de mi trayectoria. Reconozco su infinita bondad y agradezco por la oportunidad de alcanzar este hito en mi vida.

A mi amada madre, mi roca inquebrantable, le debo un reconocimiento eterno. Su apoyo incondicional y su amor han sido mi fuente de fortaleza durante cada día de mi travesía universitaria. Agradezco por su sacrificio y dedicación, que han sido la luz que ilumino mi sendero hacia el éxito.

No puedo dejar de reconocer el invaluable aporte de mi director de trabajo de integración curricular, al Ing. Marco Suing Ochoa. Su guía experta, paciencia, conocimientos y su dedicación inquebrantable fueron pilares fundamentales en la culminación exitoso de este proyecto.

Agradezco a cada uno de los distinguidos docentes de la carrera de Ingeniería en Telecomunicaciones, cuyo compromiso y dedicación fueron ejemplares en mi formación académica, mi gratitud a la Universidad Nacional de Loja por brindarme el privilegio de enriquecerme en sus instalaciones, cultivando mi talento y forjando mi profesión.

Quiero expresar mi profundo agradecimientos a cada uno de mis compañeros con los que hemos compartido en esta etapa universitaria, especialmente aquellos amigos de los que nuestro vinculo trasciende lo académico para convertirse en una hermandad de apoyo mutuo, con ustedes se compartió risas, desvelos, proyectos y sueños, Ibeth R, Karla P, Brayan A, Steven D, por su compañerismo inquebrantable a lo largo de este arduo pero gratificante viaje hacia la realización de nuestros sueños personales y metas profesionales.

Expreso mi profunda gratitud a cada persona que creyó en mí y me brindo su apoyo incondicional en la búsqueda y realización de mis sueños lejos de casa. Vuestra confianza y aliento fueron combustible para mi determinación y perseverancia.

Diana Marisol Lozano

Índice de contenidos

Índice de Tablas

Índice de Figuras

Índice de Anexos

1. Título

Desarrollo de una propuesta de diseño basada en redes definidas por software (SDN) para la red de datos de la Universidad Nacional Loja, campus la Argelia, ciudad de Loja

2. Resumen

El presente trabajo, se centra en el desarrollo de una propuesta de diseño de red basada en redes definidas por software (SDN- *Software Defined Networking*), para la red de datos del campus «La Argelia» de la Universidad Nacional de Loja. Inicialmente, se abordan aspectos relevantes relacionados con las redes de información, destacando el progreso alcanzado hasta el momento, luego se presentan los objetivos de la investigación, alcance del tema y otras consideraciones importantes. Asimismo, se presentan los fundamentos teóricos explorados sobre la conceptualización de redes definidas por software, haciendo especial énfasis en herramientas de software libre. Estas últimas que permiten la simulación y administración de la red de datos de la Universidad Nacional de Loja, utilizando protocolos controladores, herramientas y aplicaciones eficientes. Específicamente, GNS-3 nos permite realizar el análisis del funcionamiento de la red de datos bajo una arquitectura tradicional, y para la infraestructura SDN, mediante Mininet, Open Virtual Switchs (OVS) y el controlador OpenDayLight. Además, para la optimización de la red de datos de la UNL, se simula y valida una nueva arquitectura basada en SDN. En consecuencia, se exponen las pruebas y resultados obtenidos de la simulación, con el fin de comparar los beneficios que ofrece la arquitectura SDN con la implementación actual. Finalmente, este proceso permite validar los planteamientos teóricos y la investigación del tema, para brindar una evaluación integral del impacto y la viabilidad de la propuesta de diseño basada en SDN.

Palabras clave: *sdn, red de datos, optimización, GNS-3, mininet*

Abstract

The present work focuses on the development of a design proposal based on Software Defined Networking (SDN) for the data network of the Universidad Nacional de Loja campus " La Argelia". Initially, relevant aspects related to information networks are addressed, highlighting the progress achieved so far. The research objectives, scope of the topic, and other important considerations are presented. The explored theoretical foundations revolve around the conceptualization of software-defined networks, with a particular emphasis on open-source software tools that allow simulation and management of the Universidad Nacional de Loja data network, using controller protocols, efficient tools and applications.GNS-3 enables the analysis of the data network's operation under a traditional architecture and for the SDN infrastructure, Mininet, Open Virtual Switches (OVS), and the OpenDayLight controller are utilized. To optimize the UNL data network a new SDN-based architecture is simulated and validated. The tests and results obtained from the simulation are presented, allowing a comparison of the benefits offered by the SDN architecture compared to the current implementation. This process validates the theoretical propositions and research on the topic providing a comprehensive assessment of the impact and feasibility of the SDN-based design proposal.

Keywords: sdn, data network, optimization, GNS-3, mininet

Lista de Acrónimos

Tabla 1. Acrónimos

Inicial	Español	English
A		
AAA	Autenticación, Autorización, Acceso	Authentication,Authorization and Accounting
API	Interfaz de Programación de Aplicaciones	Application Programming Interface
B		
C		
CLI	Interfaz de Línea de Comando	Command Line Interface
D		
DC	Centro de Datos	Data Center
E		
F		
G		
GUI	Interfaz de Usuario Gráfica	Graphical User Interface
H		
HTML	Lenguaje de Marcado de texto	Hypertext Markup Language
HTTP	Protocolo de Transferencia de Hipertexto	Hypertext Transfer Protocol
I		Internetwork Operating System
IOS		
IP	Protocolo de internet	Internet Protocol
ICMP	Paquete del protocolo IP que guarda información del tráfico IP.	Internet Control Message Protocol
IPERF	Herramienta para hacer pruebas en redes informáticas	
J		
JSON		Javascript Object Notation
L		
M		

MD-SAL	Capa de abstracción de servicios basada en modelos	Model-Driven Service Abstraction Layer
MPLS	Cambio de etiquetas multiprotocolo	Multiprotocol Label Switching
MQTT	Transporte de telemetría de colas de mensajes	Message Queuning Telemetry Transport
N		
NETCONF	Protocolo de configuración de red	Network Configuration Protocol
NOS	Sistema operativo de red	Network Operation System
O		
ODL		OpenDayLight
OVS	Conmutador Virtual	Open Virtual Switch
ONF	Fundación de redes abiertas	Open Networking Foundation
ONOS	Sistema operativo de red abierta	Open Network Operating System
OSPF	Primer Camino Más Corto	Open Shortest Path First
P		
Q		
QoS	Calidad de Servicio	Quality of Service
R		
REST	Transferencia de estado representacional	Representational State Transfer
RFC		Request For Commen
RIB		*Routing Information Base*
S		
SSH		Secure Shell
SDN	REDES DEFINIDAS POR SOFTWARE	Software Defined Networking
SNMP	Protocolo Simple de Manejo de Red	Simple Network Management Protocol
T		
TCP	Protocolo de Control de Transmisión	Transmission Control Protocol
U		
UDP		User Datagram Protocol

V		
VM		Virtual Machine
VLAN	Virtual LAN	
W		
X		
Y		
YANG		Yet Another Next Generation
Z		

3. Introducción

Las Redes Definidas por Software (RDS) en inglés conocidas como *Software Defined Networks* (SDN), surgieron como respuesta a los desafíos inherentes de las redes tradicionales en la industria de las telecomunicaciones, las cuales fueron diseñadas para funciones específicas, limitando así su capacidad de adaptación y evolución. En este contexto, la presente investigación se centra en la necesidad de adoptar nuevas tecnologías como las SDN, que representan una alternativa viable para mejorar la conectividad.

Las SDN ofrecen un mayor control sobre los recursos de la red y presentan una visión integral, ya que sus dispositivos y aplicaciones son programables. Esta característica facilita la solución de problemas relacionados con enrutamiento, topología, estado y comportamiento de las redes adyacentes. Además, cuentan con una infraestructura idónea para la eficiente administración de datos.

En la actualidad, las redes computacionales requieren la integración de diversos dispositivos, lo que conlleva a dificultades en su administración y gestión. A lo largo del tiempo, las redes informáticas han sido testigos de grandes cambios tecnológicos, pero el diseño del esquema de red ha permanecido inalterado. Las SDN representan un cambio significativo al enfocarse en que sea el software, en lugar del hardware, el responsable de controlar la red. Este enfoque asegura que el avance de las redes se realice al ritmo del desarrollo del software.

El análisis detallado de las SDN contribuye de manera sustancial a mejorar la eficiencia de las redes, abordar desafíos de seguridad, promover el desarrollo de estándares y mejores prácticas, y ofrecer soluciones innovadoras para optimizar la utilización de los recursos de red, mejorar la escalabilidad y aumentar la confiabilidad de las redes SDN.

Por lo tanto, para las facultades académicas de la Universidad Nacional de Loja, la investigación y futura implementación de este tipo de red se posicionan como aspectos de vital importancia, ya que permitirán mantenerse a la vanguardia en el ámbito tecnológico y contribuir al avance continuo en el campo de las redes de comunicación

El objetivo general de investigación se enfoca en desarrollar una propuesta de Diseño basada en SDN para la red de Datos de la UNL, campus la Argelia. Así mismo los objetivos específicos se detallan a continuación:

- ✓ Investigar los conceptos fundamentales, de la arquitectura, infraestructura, protocolos y demás componentes que se emplean en las SDN.
- ✓ Diseñar una propuesta de optimización basada en SDN sobre el estado actual de la red de datos de la UNL.
- ✓ Evaluar la propuesta planteada mediante simulaciones y criterios teóricos técnicos, considerando tanto la infraestructura de red actual como la implementación de SDN.

Para alcanzar este propósito, se llevará a cabo un análisis detallado de la topología actual de la red, utilizando simulaciones cuando sea pertinente mediante el empleo de software libre. Además, se realizarán análisis teóricos y técnicos para comprender a fondo los aspectos críticos que afectan la gestión, disponibilidad y confiabilidad de la infraestructura de red en las facultades académicas de la Universidad Nacional de Loja, específicamente en el campus la Argelia.

El resultado esperado de esta investigación no solo se traduce en la identificación de soluciones efectivas para mejorar la infraestructura de red, sino que también busca proporcionar recomendaciones concretas y prácticas. Este enfoque integral permitirá a las facultades académicas optimizar la gestión de sus recursos, garantizar una mayor disponibilidad de servicios y fortalecer la confiabilidad de su infraestructura de red.

En busca de alcanzar estos objetivos, se plantea abordar una serie de interrogantes clave mediante una cuidadosa investigación. Se pretende dar respuesta a las siguientes preguntas de investigación:

- ¿Es posible que la red de datos de la UNL, bajo la tecnología de SDN permita optimizar la gestión de los dispositivos de red mediante el control centralizado?
- ¿Cuáles son los conceptos fundamentales sobre SDN, y como pueden ser aplicados de manera efectiva en el contexto de la red de datos de la UNL?
- ¿Cuál es el desempeño actual de la red de datos en el campus la Argelia de la UNL?
- ¿De qué manera se puede diseñar una propuesta de optimización basada en SDN que aborde los desafíos identificados en la red de datos de la UNL, considerando su actual arquitectura de red tradicional?

- ¿Cuál sería el impacto de mejorar la red de datos de la UNL, mediante la propuesta de optimización al realizar una comparación con la infraestructura de red existente?

4. Marco teórico

En este capítulo se van a revisar los fundamentos que sustentan los procedimientos y resultados obtenidos en la presente investigación. Es un marco teórico que revisa conceptos básicos y otros conceptos específicos necesarios para la presente investigación.

4.1. Redes Definidas por Software

4.1.1. Definición

Organización de Redes Abiertas (ONF - *Open Networking Fundation*), es una organización que fue creada sin fines de lucro, es dedicada únicamente para el desarrollo de los distintos estándares abiertos, y estudios sobre SDN. Es por lo que ONF (2024), más formalmente define que:

> Una red definida por software es una arquitectura emergente que es dinámica, administrable, rentable, y adaptable, lo cual la hace ideal para naturaleza dinámica y de alto consumo de ancho de banda de las aplicaciones de hoy en día. Esta arquitectura separa las funciones control y de envío de paquetes de la red, permitiendo que el control de la red sea directamente programable y que la infraestructura subyacente pueda ser abstraída para las aplicaciones y los servicios de red.

Lo que conduce a que en la actualidad la red SDN sea una alternativa para reducir las dificultades en las empresas, permitiendo que estas puedan controlar el despliegue de recursos de infraestructura. (Guanoluisa, 2019).

4.1.2. Arquitectura

En el ámbito de las redes, dos planos fundamentales coexisten para garantizar el flujo eficiente de datos y el control preciso de la red: el plano de datos y el plano de control. Salazar (2021), establece que el plano de control y los planos de datos son conceptos que se definen como:

El **Plano de Datos**, también conocido como plano de reenvío (*forwarding*) o plano de usuario, se encuentra implementado físicamente en el equipo de red. En este plano, el flujo de datos llega y sale del dispositivo siguiendo la dirección del destino a través de una interfaz específica. Y, por otro lado, el **Plano de Control** reside en el software, específicamente en el CPU del equipo. Su función principal es recopilar información de control, proveniente de distintos protocolos, para poder construir una tabla de

enrutamiento (RIB - *Routing Information Base*). Esta tabla indica al equipo la interfaz de salida que debe tomar el flujo de datos proveniente del plano de datos.

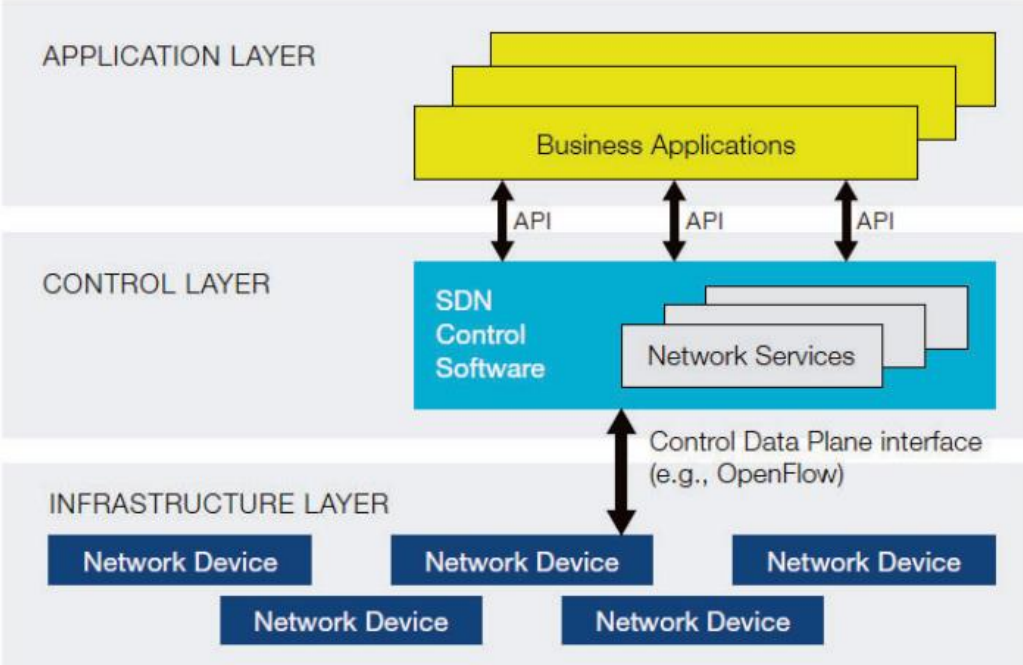
Es necesario señalar que, en dispositivos convencionales, ambos planos (control y datos) coexisten en el mismo dispositivo. Sin embargo, la arquitectura de SDN se basa en la separación del plano de control del de datos, empleando controladores basados en software, los cuales se encargan de gestionar el flujo de datos entre el controlador y los conmutadores (switch) (Alava & Paladines, 2020).

SDN se componen principalmente de 3 capas principales, en las que según Cáceres y Casilimas (2022), son:

- ✓ **Capa de Infraestructura:** los conmutadores y enrutadores físicos de la red del centro de datos se encuentran en esta capa. Dichos dispositivos son muy importantes ya que se encargan de la supervisión de las funciones cruciales de reenvió y procesamiento de datos, así como la recopilación de información vital, como el uso y topología de red, que posteriormente transmiten a la capa de control.
- ✓ **Capa de control:** es la encargada de gestionar las políticas y el flujo de trafico de la red. Su centro es un controlador SDN, que se encuentra conectado a través de Interfaces de Aplicación Programables (API – *Application Programming Interfaces*), descendente para los requisitos enviados por la capa de aplicación, y la API ascendente los transmite a la infraestructura de red actual.
- ✓ **Capa de aplicación:** Consiste en un conjunto integral de aplicaciones y funciones de red diseñadas para potenciar y simplificar el rendimiento de las redes. Incluye controladores de optimización, balanceo de carga, firewalls de aplicación, entre otros. En las redes convencionales, estas funciones suelen depender de dispositivos específicos, mientras que en SDN, el controlador desempeña el papel central en la gestión.

En la Figura 1, se puede visualizar estas tres capas principales mencionadas con anterioridad.

Figura 1.
Arquitectura de SDN



Fuente: Alava & Paladines, 2020

4.1.3. Componentes SDN

SDN cuenta en su infraestructura con componentes principales, que son esenciales para el funcionamiento correcto:

- ✓ **Controlador SDN:** Es el centro de todo, es el encargado del establecimiento de las políticas de enrutamiento. Según Aguirre y Crespo (2021), «es una entidad de software que tiene control exclusivo sobre un conjunto abstracto de recursos del plano de datos y donde se generan las políticas de información de tratamiento».
- ✓ **Dispositivos y Equipos de red:** se ubican en el plano de datos de una red SDN. Estos pueden ser físico o virtuales.
- ✓ **NOS (Network Operating System):** el sistema operativo que se utilizará para la implementación de la red, la misma puede ser de código abierto o de propietario.
- ✓ **Interfaces:** nos permiten establecer el protocolo para comunicaciones entre los equipos de red, así también APIs necesarias.

4.1.4. Interfaces

El controlador desempeña un papel crucial al segmentar las dos interfaces fundamentales que constituyen una arquitectura SDN, según señala (Santisteban, 2020), la **interfaz southbound (SBI),** se encarga de facilitar la comunicación entre los dispositivos de red y el controlador, mientras que la **interfaz northbound (NBI),** se encarga de informar al controlador sobre aplicaciones externas, también denominadas APIs. Es pertinente destacar que cada una de estas interfaces establece este intercambio de información utilizando protocolos específicos asociados con el entorno SDN.

En la Tabla 2, se describe el concepto y algunos protocolos utilizados en las interfaces para establecer comunicaciones.

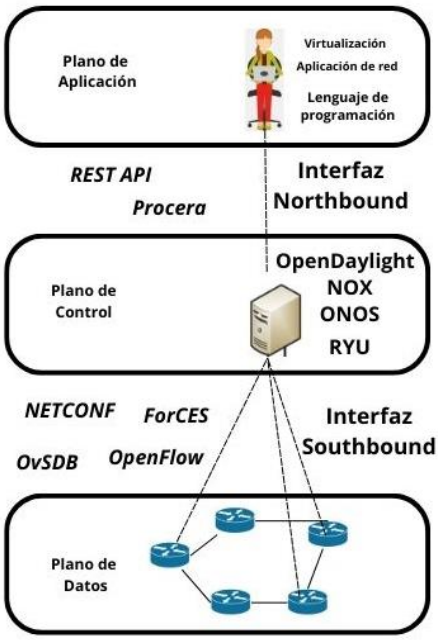
Tabla 2.
Interfaces NBI Y SBI

Interfaces	Descripción
Northbound Interface (NBI)	Es un conjunto de protocolos que facilita la comunicación entre diversas aplicaciones para supervisar tanto los dispositivos de red como el controlador. Esta API posibilita que la información de control de la red sea accesible, abstrayendo la complejidad inherente a los dispositivos de red. Existen varios protocolos Northbound que varían según la infraestructura utilizada; por ejemplo, en Cisco Systems, se implementa el protocolo de tipo RESTCONF, mientras que en entornos como OpenDayLight (ODL) además del protocolo RESTCONF, se emplean NETCONF y AMQP (Advanced Message Queing Protocol), como protocolos alternativos para establecer la comunicación. Sin embargo, es crucial tener en cuenta la falta de estandarización en dichos protocolos.
Southbound Interface (SBI)	Este conjunto de protocolos facilita la comunicación entre los elementos del plano de control y los equipos en el plano de datos. Aquí es donde se programan las instrucciones de reenvío desde los dispositivos de red. Protocolos como NETCONF, ForCES, OvSDB, OpenFlow, son algunos que destacan por su interoperabilidad y estandarización.

Fuente: Elaboración Propia

En la figura 2 se desglosa la estructura de SDN, comenzando desde el nivel superior que abarca el plano de aplicación, que incluye elementos como virtualización, aplicación de red y lenguaje de programación. Luego, en el plano intermediario, se sitúa el controlador (opendayligth, ryu, onos) encargado de facilitar la comunicación mediante la interfaz northbound, que puede incluir protocolos como rest api y procera. Este controlador se comunica con el plano inferior, conocido como plano de datos, a través de la interfaz southbound utilizando protocolos como netconf,forces y openflow. Este diseño jerárquico y modular permite una gestión eficiente y flexible de la red.

Figura 2.
Arquitectura Interfaces



Fuente: Elaboración Propia

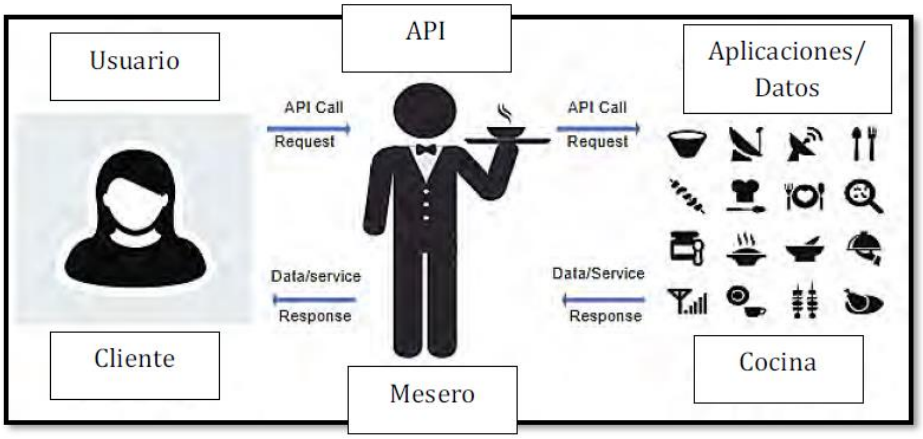
4.1.5. APIs (Application Programming Interfaces)

Los dispositivos inteligentes aprovechan extensamente los formatos de datos para facilitar la transferencia de información y llevar a cabo acciones automatizadas. Esto es posible gracias a las Interfaces de Aplicación Programables (API), un tipo de software que habilita a otras aplicaciones para acceder a sus datos y entablar interacciones con ellos. En esencia, una API posibilita la definición de un conjunto de reglas que caracterizan una aplicación, permitiéndole interactuar con otra.

En la figura 3, se visualiza el esquema de un API, en el que podemos apreciar el funcionamiento de una interfaz de Aplicación Programable, el cual sería como un

intermediario entre el usuario y el pedido (request) de información o interacción con los datos (Salazar, 2021)/Cisco NetCad.

Figura 3.
Esquema de un API



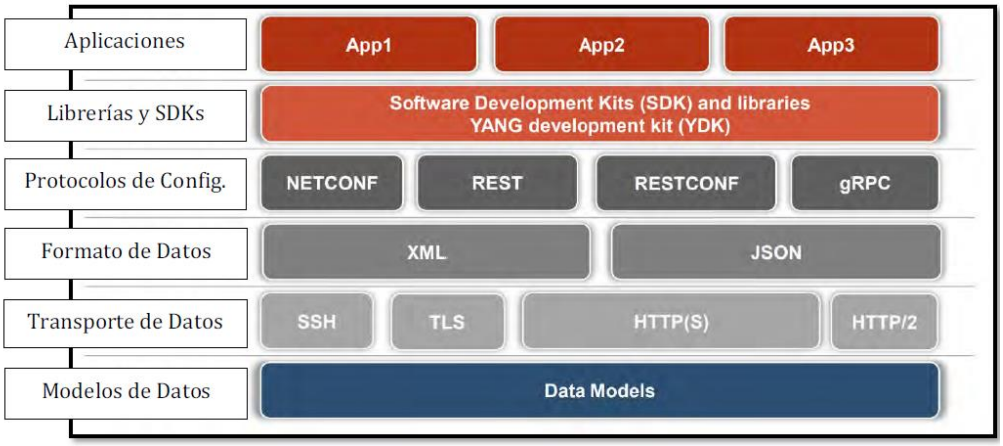
Fuente: Salazar, 2021

Según Salazar (2021), en la era de la programabilidad en la que nos encontramos, visualiza el entorno conformado por:

- **Formato de datos:** se pueden intercambiar en una infraestructura de red estructuradamente, por ejemplo: XML,YAMI,JSON.
- **Modelos de datos:** permiten estandarizar el intercambio programático, describiendo el conjunto de datos, por ejemplo: YANG, SAL.
- **Mecanismo de transporte:** necesario contar con un método seguro entre todos los actores que intercambian información, datos y ordenes en el networking moderno, por ejemplo: SSH, TLS o HTTTPS.
- **APIs**: que sirven como intermediario entre la administración, gestión y el hardware/software subyacente, por ejemplo: REST, RESTCONF y tipo NETCONF.

En la figura 4 se presenta la jerarquía mencionada:

Figura 4.
Stack para programabilidad en las redes



Fuente: *Cisco,2023*

4.1.6. OVS *(Open VSwitch)*

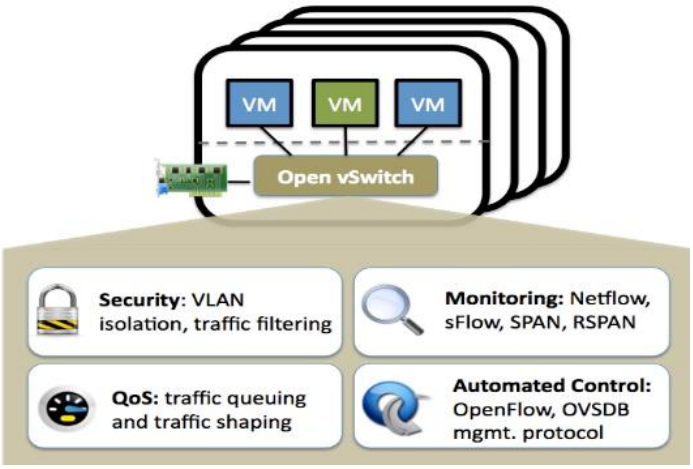
Open vSwitch es un conmutador virtual multicapa con licencia de código abierto bajo el estándar Apache 2.0. Su diseño se orienta hacia la producción de alta calidad, destacándose por su capacidad para posibilitar la automatización masiva de redes a través de extensiones programáticas mediante diversos interfaces y protocolos.

Este conmutador virtual no solo se utiliza en múltiples productos, sino que también opera en diversos entornos de producción. Su funcionamiento y las implementaciones de nuevas versiones se someten a un riguroso proceso que implica cientos de pruebas a nivel de sistema y miles de pruebas unitarias. Este enfoque asegura la estabilidad, confiabilidad y rendimiento óptimo del OVS en contextos reales de uso, representado en la Figura 5 (Open VSwitch, s.f.).

Escrito principalmente en lenguajes C++ y Python, OVS es independiente del módulo kernel de Linux y ofrece despliegue en arquitecturas de clúster para garantizar una alta disponibilidad. Su funcionamiento se basa en permitir una programabilidad y flexibilidad notable.

Open vSwitch presenta un sólido respaldo para funciones clave, como VLAN802.1Q, LACP para puertos integrados, QoS y protocolos de enrutamiento dinámico. Su versatilidad radica en la capacidad de establecer conectividad a nivel de enlace y nivel de red entre nodos virtuales que se ejecutan en un mismo anfitrión.

Figura 5 .
Funcionalidades de Open vSwitch

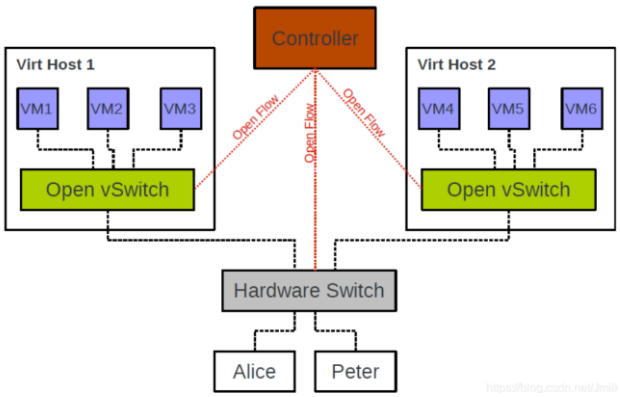


Fuente: Open VSwitch, s.f.

Open vSwitch opera mediante diferentes tipos de flujos para distintos propósitos, siendo los flujos OpenFlow los más relevantes. Estos flujos proporcionan flexibilidad y programabilidad al switch virtual, mientras que los controladores OpenFlow los utilizan para definir las políticas del conmutador. Los Flujos de OpenFlow admiten prioridades y se gestionan en múltiples tablas, que se detalla más adelante.

La Figura 6 se visualiza la interacción entre el controlador SDN, que puede ser OpendayLight u otro similar, con el Open vSwitch en un entorno de virtualización, bajo el protocolo de comunicación OpenFlow, para administrar de forma remota, gestiona dos conjuntos de OVS en nodos de host, cada uno conectado a 3 redes virtuales, que se crean aisladas dentro de un entorno compartido. Estos OVS están conectados a un dispositivo hardware switch, que tiene 2 entradas (Alice y Peter) que pueden comunicarse con las redes virtuales proporcionadas por OVS (Graf, 2013).

Figura 6.
Esquema de Funcionamiento OVS



*Fuente:*Graf, 2013

4.1.7. Protocolos SDN

Examinaremos algunos de los protocolos más destacados utilizados en SDN, centrándonos en la interfaz **Southbound (SBI),** que es la que nos permite que el plano de control establezca comunicaciones con el plano de datos. Su principal función radica en su capacidad para simplificar la tarea del controlador al configurar o controlar los procesos del switch (Aguirre & Crespo, 2021).

De entre los distintos protocolos de software libre establecidos en **SBI APIs,** existe una gran variedad, considerando sus características y especificaciones se consideró los siguientes:

- **NETCONF (*Network Configuration Protocol*)**

Es un protocolo estandarizado de configuraciones de redes, desarrollado por IETF en 2006, como respuesta a la necesidad de simplificar la configuración de dispositivos de red.

Teniendo en cuenta que el diseño de redes frecuentemente implica la utilización de equipos provenientes de diversos fabricantes, este protocolo aborda dicha diversidad al unificar la gestión de estos dispositivos. Esto posibilita la ejecución de acciones específicas en la red de manera más eficiente y coherente.

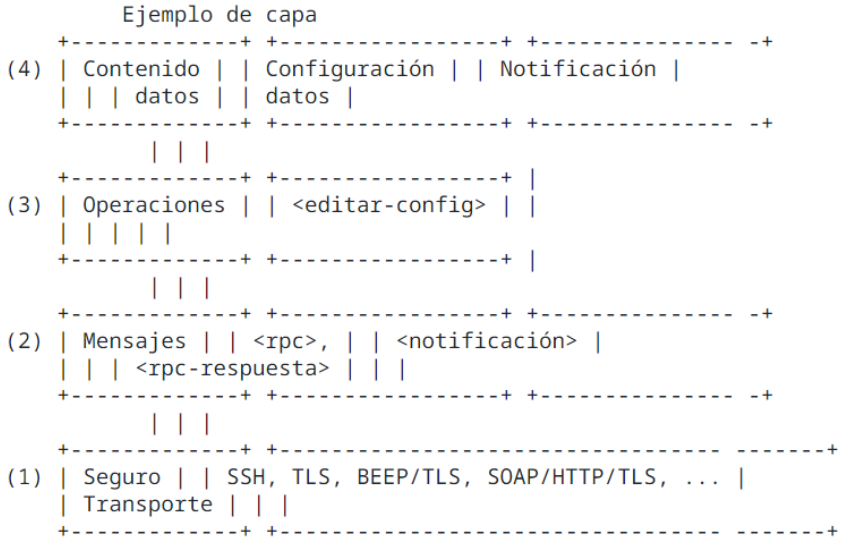
Arquitectura

Este protocolo se sirve de un mecanismo fundamentado en RPC (Remote Procedure Call), para facilitar la interacción entre un cliente y un servidor de manera remota. Su función principal radica en mantener la comunicación durante la sesión establecida mediante NETCONF. En este contexto, la sesión se establece entre un

administrador de red (cliente) y un dispositivo de red (posiblemente un servidor). Es crucial destacar que el dispositivo debe ser capaz de admitir al menos una sesión NETCONF y permitir múltiples sesiones.

Asimismo, este protocolo mantiene atributos, los cuales pueden ser de naturaleza global o específica. Para una mejor comprensión, se puede dividir en cuatro capas, como se ilustra en la figura 7.

Figura 7.
Arquitectura de NETCONF



Fuente: RFC 6241 (Español)

Es relevante destacar que al llevar a cabo cualquier implementación con NETCONF, es imperativo que se respalde el protocolo de transporte SSH. Además, la codificación se lleva a cabo en formato XML, ya que este formato proporciona la capacidad de lectura, almacenamiento y manipulación mediante diversas herramientas. El sistema puede incluir documentación almacenada en varias bases de datos, abordando temas como topologías, enlaces, clientes y servicios. No obstante, es importante señalar que carece de soporte para algunos simuladores de red como Mininet, Estinet, entre otros, “los mismos que serán abordados en detalle posteriormente,” lo cual representa una limitación significativa para la realización de diversos estudios en el ámbito de SDN (RFC 6241).

- **OvSDB (*Open vSwitch Database*)**

Protocolo basado en Open vSwitch, utilizado para gestionar la configuración y el estado del mismo, se usa diferentes tipos de flujos para diferentes propósitos, cuenta con

un sistema de base de datos accesible en la red, lo que permite mantener la sincronización para la administración de la base de datos que almacena la configuración de OVS.

Esquemas

Conforme se detalla en el RFC 7047, OvSDB emplea un formato JSON que posibilita la comunicación entre sus clientes y servidores. Un esquema de OVS consta de tres identificadores cruciales. En primer lugar, se encuentra el nombre denominado JSON-RPC, el cual sirve para identificar la base de datos, es decir, el esquema utilizado para el nombre.

Además, dicho esquema posee una versión que gestiona la numeración de versiones, aunque algunos esquemas de antigüedad considerable pueden carecer de una versión específica. En el contexto de OVS, se hace mención a una suma de verificación que se utiliza para mantener la coherencia del esquema con una versión actualizada, garantizando así la conformidad con la política de numeración de versiones. Este enfoque asegura que el esquema se mantiene al día y cumple con las directrices establecidas en cuanto a versiones.

Modelos de servicios

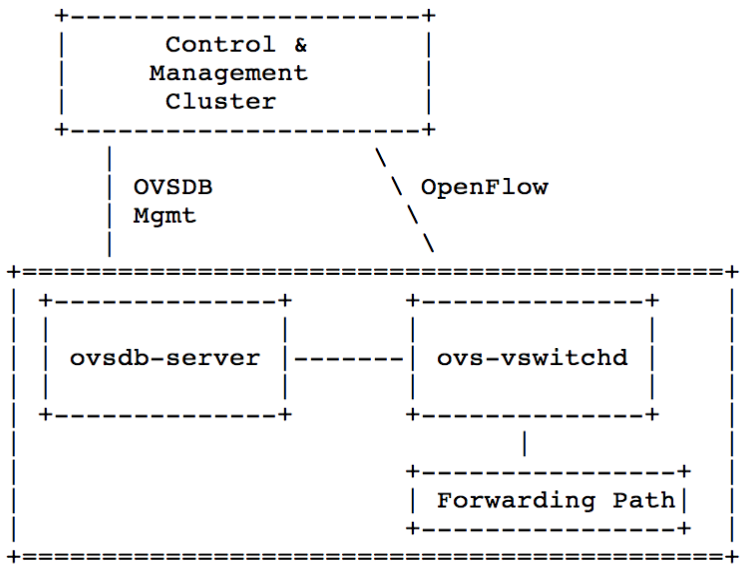
Se requiere mantener cuatro modelos de servicio específicos para bases de datos, a saber: independiente, de respaldo activo, de retransmisión y en clúster. Estos modelos posibilitan diversas combinaciones en cuanto a coherencia, disponibilidad y tolerancia de partición, abordando diferentes aspectos de rendimiento. Tanto la base de datos independiente como la de respaldo activo comparten un formato en disco, mientras que las bases de datos agrupadas utilizan un formato distinto. Sin embargo, es destacable que los programas OvSDB están diseñados para operar eficientemente en ambos formatos.

El modelo de retransmisión no almacena datos en el disco, y el modelo de servicio en clúster implica la presencia de 3, 5 o más servidores de bases de datos distribuidos en distintos hosts. Estos servidores pueden tolerar ciertas fallas de manera aceptable. Además, mediante un comando específico, es posible crear una base de datos vacía o copiar el contenido de una base de datos independiente a una nueva.

Es esencial tener en cuenta y subrayar que OvSDB tiene la capacidad de compactar un archivo de base de datos, lo cual implica la posibilidad de reemplazarlo con nuevas versiones que contengan únicamente la información actualizada, en la Figura 8, se presenta un diagrama que describe la relación complementaria pero claramente

separada entre los procesos de control y gestión. En una implementación de OVS, existe un proceso dedicado, «ovsdb-server», creado para aceptar cambios de configuración desde un controlador, este proceso que se asemeja a la inserción de datos para generar una nueva transacción. Cabe mencionar que OvSDB y OpenFlow tienen fortalezas complementarias, pero no se debe asumir que se necesitan mutuamente.

Figura 8.
Modelos de Servicio



Fuente: OvSDB, 2023

OVSDB permite la configuración de políticas QoS, y es por esto que permite el uso combinado con OpenFlow, y un soporte en la virtualización con NS-3, Mininet, GNS-3, aunque cabe mencionar que su flexibilidad es muy poca ya que mantiene su propia estructura. (Ovsdb,2023.)

- **ForCES (*Forwarding and Control Element Separation*)**

Un protocolo compuesto por Elementos de Control (CE-*Control Element*) y Elementos de Reenvío (FE- *Forwarding Element*), gestiona funciones después de establecer la asociación de comunicación entre CE y FE. Se configura como un protocolo de tipo maestro-esclavo, donde los FE actúan como esclavos y los CE como maestros. Este protocolo puede ser singular o compuesto por múltiples protocolos que colaboran simultáneamente.

El propósito central de ForCES radica en establecer un marco y protocolos asociados para estandarizar el intercambio de información entre el control y el reenvío.

Esto posibilita considerar estos componentes como entidades esencialmente separadas, lo que facilita la interoperabilidad entre expertos en componentes, brindando así flexibilidad y diversas opciones de diseño.

Arquitectura Maestro-Esclavo

FE es el encargado de reenviar paquetes en el dispositivo de red, diseñado para ser implementado en hardware, según las tablas de reenvío, y **CE,** es el encargado de tomar decisiones de control en el dispositivo de red, como la gestión de políticas de red, reglas de reenvío, entre otros.

A pesar de la documentación limitada, el RFC3746 destaca aspectos cruciales para un elemento de red, incluyendo uno o más FE y CE, con un administrador opcional. La separación e interoperabilidad entre CE y FE debe considerarse para una estandarización efectiva y práctica. Aunque presenta limitaciones al ser específico del controlador y solo permite implementaciones con interfaz CE-FE, carece de soporte para componentes externos (RFC 3746).

- **OpenFlow**

Protocolo estandarizado diseñado para la implementación de SDN, con la meta de lograr que la red sea programable de manera centralizada. Esto posibilita la programación de las tablas de flujo en diversos routers y switches.

Es importante destacar que el administrador de red tiene la capacidad de segmentar el tráfico en diferentes flujos para optimizar el procesamiento de paquetes. OpenFlow ofrece la posibilidad de introducir nuevas estrategias de enrutamiento, control de tráfico, direccionamiento, entre otros aspectos.

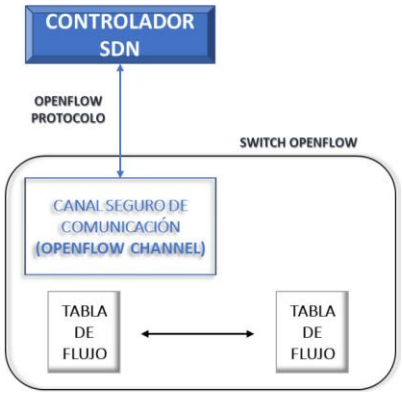
En la práctica, OpenFlow emplea diversos flujos con reglas predefinidas, las cuales pueden ser programadas de forma estática o dinámica mediante un software de control. Este último facilita la identificación del tráfico de la red, permitiendo la actualización de cambios en tiempo real en los niveles de aplicación, usuario y sesión, respectivamente (Guanoluisa, 2019).

Aguirre y Crespo (2021), menciona que la estructura central de OpenFlow se basa en un Switch Ethernet, que facilita la comunicación lógica entre un switch y el controlador SDN a través de un canal seguro. Esta característica es clave ya que posibilita la interoperabilidad entre dispositivos de diferentes fabricantes, permitiendo que cualquier equipo físico o virtual compatible con OpenFlow sea administrado por el controlador. En

cuanto a los componentes esenciales de OpenFlow, según lo definido por la ONF (Organización de Redes Abiertas- Open Networking Foundation), representando en la figura 9 incluyen:

- **Tablas de flujo:** Establecen cómo debe procesarse un flujo específico de paquetes.
- **Canal seguro:** Garantiza la comunicación segura entre el switch y el controlador a través del protocolo OpenFlow, respaldando así la seguridad necesaria.
- **Controlador SDN:** Responsable de gestionar toda la red de manera centralizada y encargado de la administración de las tablas de flujo.

Figura 9.
Esquema Protocolo Openflow



Fuente: Chafloque, 2018

Las diversas posibilidades ejecutables en las variadas reglas preconfiguradas incluyen la autorización, el enrutamiento o la eliminación de flujos. Un switch OpenFlow puede clasificarse como OpenFlow dedicado o OpenFlow hibrido. Un **switch dedicado** transmite paquetes exclusivamente mediante las tablas de flujo, a diferencia de un **switch hibrido**, que es un dispositivo capaz de conmutar paquetes utilizando la lógica tanto de un router como de un switch convencional, empleando su propio plano de control (Chafloque, 2018).

Open Networking Foundation, (n.d.) establece que una tabla de flujo en un switch OpenFlow se construye al definir los valores de uno o más campos en la cabecera del paquete que el switch está procesando. Estos campos, que se presentan en la figura 10, constituye un conjunto de siete entradas, cada una desempeñando una función específica:

Figura 10.
Tabla de flujo

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie	Flags

Fuente: Intriago, 2017

El protocolo OpenFlow gestiona tres categorías diferentes de mensajes: aquellos que van del controlador al switch, mensajes simétricos y mensajes asíncronos. Estos mensajes establecen cómo debe reaccionar el dispositivo en diversas situaciones y cómo responder a los comandos del controlador. Los mensajes del **controlador al switch** son iniciados por el controlador, y se emplean para gestionar o inspeccionar directamente el estado del switch. Por otro lado, los **mensajes asíncronos** son iniciados por el switch y sirven para mantener al controlador actualizado sobre los eventos de red y los cambios en el estado del switch. Los **mensajes simétricos** pueden ser iniciados tanto por el dispositivo como por el controlador sin necesidad de una solicitud previa (Intriago, 2017).

Cumpliendo con las necesidades del proyecto SDN, se seleccionó el protocolo OpenFlow, teniendo en cuenta sus múltiples ventajas que sobresalen de los otros protocolos.

Siguiendo con la arquitectura de SDN, es esencial destacar la interfaz **Northbound (NBI) API** que nos permite visualizar a los equipos interconectados en la red, Aguirre y Crespo (2021), nos explica que API NBI «permite que las aplicaciones tengan control de la red y proporciona una interfaz común entre el controlador y el plano de administración», resaltando así su principal desventaja. Según Pacuar (2022), dice que «no existe una estandarización y cada fabricante selecciona su API, lo que conlleva a que se debe conocer el funcionamiento de las APIs del fabricante con el que se va a operar».

A diferencia de la interfaz SBI, donde se establecen distintos protocolos, en NBI no hay un protocolo o API de comunicación estandarizado. En este sentido, Padròn Pèrez (2020),menciona que existe uso de tecnologías estandarizadas para realizar las comunicaciones, como por ejemplo puede ser las **REST APIs,** y es la seleccionada para el proyecto debido a su enfoque flexible y eficiente para interactuar con el controlador y gestionar la red.

4.1.8. Controladores

Según Oviedo et al. (2021), mencionado por Escobar (2015), definen que «el controlador es el núcleo central de la arquitectura SDN». Es importante mencionar que

en la actualidad existen diversos tipos de controladores desarrollados para SDN, que cuentan con distintas características que los diferencian a cada uno.

- **NOX**

El producto, creado por Nicira Networks en colaboración con el protocolo OpenFlow, se destaca como el primer controlador de código abierto en su género. Entre sus atributos más destacados se encuentra el mantenimiento de una interfaz de programación de aplicaciones (API) en C++. Además, presenta una notable velocidad y asincronía que facilitan el manejo de la entrada y salida de flujos de datos en cualquier momento.

Este controlador puede ser implementado en diversos sistemas operativos Linux, incluyendo varias versiones, así como en distribuciones como Debian, Red Hat y CentOS. Dispone de dos versiones disponibles: Nox Classic, que fue desarrollado en Python y C++, aunque se encuentra en desuso y ya no se recomienda para nuevas implementaciones (Gómez, 2016).

La versión más reciente de este software ha sido desarrollada exclusivamente en C++, lo que resulta en un rendimiento considerablemente más rápido. Esta actualización se benefició de continuas mejoras, contando con el respaldo de varios equipos desarrollados.

Aunque presenta un menor modularidad, y documentación limitada, cuenta con una compatibilidad con protocolos de red como OpenFlow, NETCONF, ForCES entre otros, y su enfoque se centra en la supervisión de eventos proporcionando una plataforma eficiente para la programación de tareas.

A pesar de sus limitaciones en escalabilidad para cambios complejos, NOX es compatible con multiprocesos. Su rendimiento podría no considerarse elevado en situaciones que implican procesos complejos, pero gracias a su simplicidad, demuestra eficacia en términos de velocidad (Ruipérez, 2021).

- **ONOS (*Open Network Operating System*)**

Diseñada para adaptarse a entornos de SDN y satisfacer diversas necesidades, ONOS proporciona flexibilidad para la implementación de servicios de red dinámicos con interfaces programables simplificadas. ONOS gestiona la configuración como parte integral del control en tiempo real de la red, eliminando procesos innecesarios. Al

trasladar la inteligencia al controlador de la nube de ONOS, se fomenta la innovación para los usuarios finales, permitiendo la introducción de nuevas aplicaciones de red sin afectar los sistemas del plano de datos.

ONOS cuenta con una gran comunidad de desarrolladores y usuarios, además de que sea un proyecto de código abierto distribuido bajo la licencia de Apache 2.0, mantiene una plataforma web y un conjunto de aplicaciones que actúan como un controlador SDN distribuido, modular y extensible (ONOS, n.d.).

Arquitectura ONOS

ONOS cuenta con una arquitectura que separa el plano de control del plano de datos en las redes, permitiendo la gestión centralizada y programable de los dispositivos de red. Este proyecto está basado en Open Services Gateway Initiative (OSGi) desarrollado por Java, por lo que desde sus inicios se ha desarrollado manteniendo características en su topología, enrutamiento, interfaz gráfica de usuario web, aplicación, núcleo y administrador de dispositivos, como se muestra en la Figura 11:

Modularidad del código; este proyecto funciona con distintos componentes, mantiene funciones proporcionadas por Maven, que es una herramienta de gestión y comprensión de proyectos de software.

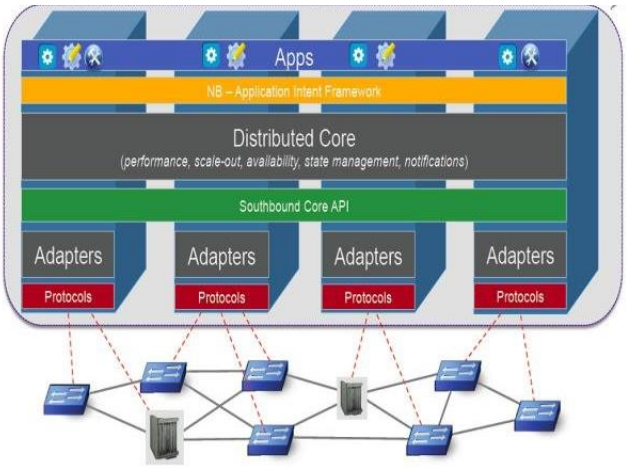
Configurabilidad; permite activar o desactivar distintas características en el tiempo de ejecución.

Separación de intereses; entre los módulos que tiene ONOS:

- Módulos orientados a la red con reconocimiento de protocolos que interactúan con la red.
- Núcleo del sistema para poder rastrear y proporcionar información sobre el estado de la red.
- Aplicaciones que consumen y actúan sobre la información proporcionada por el núcleo.

Protocolo de agnosticismo; al momento de admitir un nuevo protocolo, es posible construir un nuevo módulo orientado a la red contra SBI API.

Figura 11.
Arquitectura ONOS



Fuente: Sameer & Goswami,2018

- **RYU**

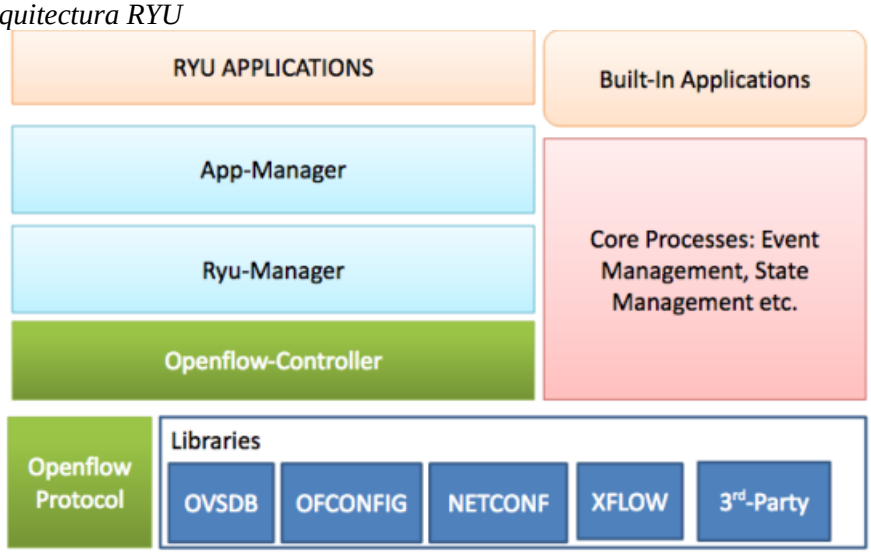
RYU representa un framework de SDN que ofrece componentes a través de una API, simplificando la creación de nuevas aplicaciones para el control y la administración de redes. Aunque su documentación sobre funcionamiento y procesos es limitada, se presenta como un controlador que es compatible con varios protocolos, incluyendo OpenFlow, Netconf y OF-config, para la gestión de dispositivos de red.

Este controlador, que admite multiprocesos, puede no ser la opción más adecuada para un rendimiento óptimo en topologías de red complejas. Sin embargo, destaca por ser de código abierto, bajo la licencia Apache 2.0. Aunque carece de interoperabilidad con equipos tradicionales, ofrece soporte para virtualización mediante Mininet, y está desarrollado en el lenguaje de programación Python. Compatible con sistemas operativos como Linux y macOS, su modularidad es moderada, lo que simplifica la sintaxis y semántica de su uso.

Arquitectura RYU

RYU, puede crear y enviar un mensaje Openflow, así como analizar y manejar los paquetes entrantes, sigue una arquitectura basada en eventos de red específicos. Permitiendo una respuesta dinámica a cambios en la red y la implementación de lógica de control personalizada, en la Figura 12, se muestra la arquitectura del *framework* del controlador:

Figura 12.
Arquitectura RYU



Fuente: SDN RYU,2022

Es fundamental destacar que este controlador dispone de una colección de bibliotecas que abarcan desde la compatibilidad con diversos protocolos en la capa de conexión hacia el sur (*SouthBound*) hasta diversas operaciones de procesamiento de paquetes de red (Ryu, 2022.).

- **ODL (*OpenDaylight*)**

Es un controlador de código abierto y modular, desarrollado por IBM y Cisco. Su finalidad principal es posibilitar el desarrollo, la administración y la implementación transparente de la tecnología SDN. ODL representa la integración de diversos proyectos, lo cual se ha traducido en un desarrollo activo con distintas versiones a lo largo del tiempo. En cada lanzamiento, se lleva a cabo la mejora o eliminación de características y componentes específicos, y para este trabajo de investigación se utilizó la versión de Lithium (Aguirre y Crespo, 2021).

Entre las herramientas que utiliza para su funcionamiento tenemos, en base a ODL (2023), las siguientes:

- **Maven:** es una herramienta de administración, crea scripts con las dependencias entre paquetes, facilitando la implementación de funciones para el proyecto.
- **Karaf:** se trata del entorno de ejecución de la plataforma ODL, cuenta con una interfaz OSGi (Open Services Gateway Initiative), que consta del conjunto de estándares abiertos que permite cargar dinámicamente paquetes

y archivos, para intercambiar información, concentrándose en toda la lógica de ODL.

- **DLUX:** es la interfaz gráfica que sirve para visualizar la topología de la red, estadísticas y gestionar las APIs. (Domínguez Pérez, 2021)
- **YANG (*Yet Another Next Generation*):** es un lenguaje de modelado de datos utilizado para modelar la configuración y los datos de estado manipulados por las aplicaciones.
- **AAA (*Authentication, Authorization and Accounting*):** controla el acceso a los recursos, aplica políticas para utilizar esos recursos y audita su uso. Pilares fundamentales para una gestión y seguridad eficaces de la red.

Arquitectura ODL

ODL cuenta con una arquitectura modular que permite la incorporación de distintos módulos y características según las necesidades específicas de implementación. Lo que nos permite la adaptación del controlador SDN a diferentes entornos, entregándonos una flexibilidad a la inclusión de servicios automatizados, además de soportar perfectamente aplicaciones de negocios utilizadas para la gestión de la red, cabe mencionar que son desarrollados por terceros y hacen uso de la API Northbound para comunicarse con el controlador.

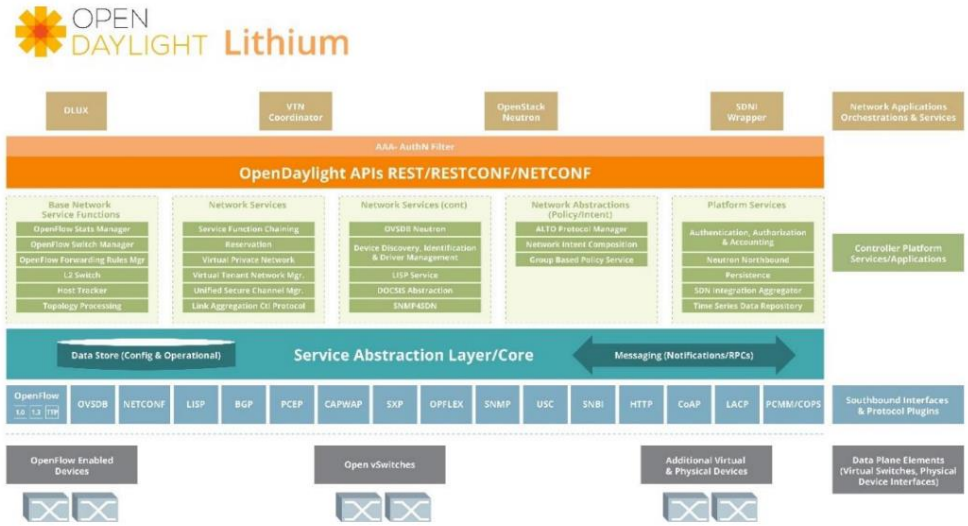
El controlador ODL admite acceso externo a aplicaciones y datos utilizando los siguientes protocolos:

- NETCONF: protocolo RPC basado en XML, que proporciona capacidades como recibir notificaciones, modificar, manipular, leer datos modelados por YANG.
- RESCONF: protocolo basado en HTTP, nos permite manipular datos modelados por YANG, utilizando XML o JSON, como lenguajes de programación, para la carga útil.

El núcleo de la plataforma se la denomina a la Capa de adaptación de Servicios basada en Modelos **(**MD-SAL/ *Model-Driven Service Abstraction Layer***)** es un componente de middleware extensible inspirado en el bus de mensajes que proporciona funcionalidad de mensajería y almacenamiento de datos basada en modelos de interfaz y datos definidos por los desarrolladores de aplicaciones (MD-SAL — ODL, 2023).

No olvidar que los dispositivos de red se pueden comunicar con el controlador utilizando *Southbound* API respectivamente, todo lo mencionado se aprecia en la Figura 13.

Figura 13.
Arquitectura ODL



Fuente: ODL,2023

Módulos de ODL

Un módulo se define como una sección independiente de un programa, desarrollada de manera autónoma. Este enfoque permite una compilación por separado, reduciendo la complejidad del programa global.

- **Odl-mdsal-clustering**: Este módulo es el encargado de permitir el funcionamiento del clúster de controladores.
- **Odl-l2switch-switch-ui:** Este es el módulo que permite la conectividad en la red, admite aprender direcciones MAC e IP, también elimina bucles por lo que no es necesario instalar módulos adicionales de STP y permite la instalación de flujos en cada conmutador basados en parámetros de tráfico en la red.
- **Odl-dlux (core, apps-nodes, apps-topology):** Estos módulos permiten utilizar la interfaz gráfica de ODL. Con las aplicaciones de nodos «**odldluxapps-nodes**» se podrán observar un listado de los dispositivos (conmutadores) detectados, así como sus flujos y conexiones. En la parte de topología se mostrará una representación gráfica de la red (Santisteban, 2020).

4.1.9. Herramientas de Emuladores/Simuladores de SDN

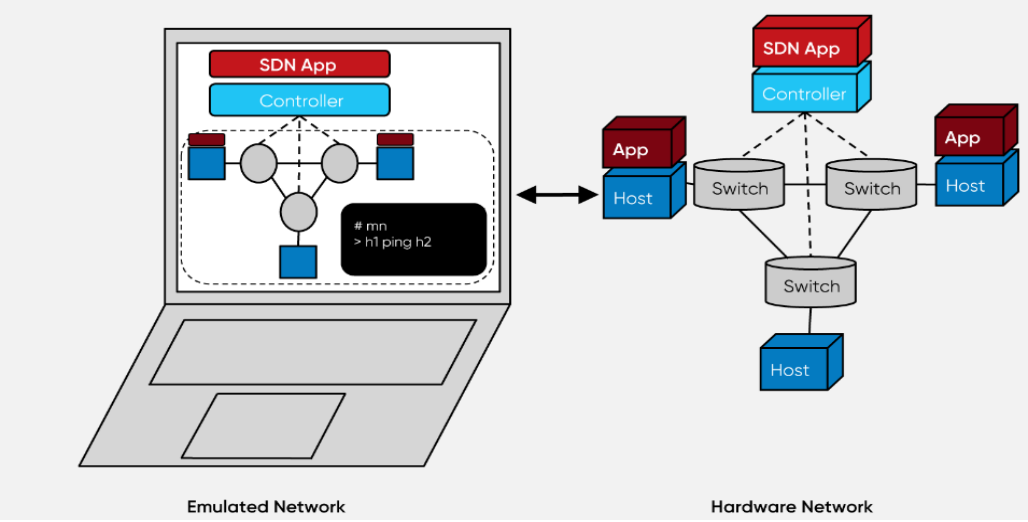
Los entornos de simulación y emulación son esenciales para el desarrollo de cualquier proyecto de redes, ya que nos permite representar un escenario parecido al de la realidad, en la que podemos visualizar el funcionamiento y poder detectar algunos errores que se puedan presentarse en la red, evitando así cualquier contratiempo antes de su despliegue real (Planas, 2016).

- **MiniNet**

Es un emulador de red, de código libre, utilizado en kernel Linux, desarrollado para crear SDN puesto que soporta OpenFlow. Este emulador nos permite crear, interactuar, personalizar diferentes topologías y componentes de redes basadas en SDN, como links, host, switches, controladores. Esta herramienta cuenta con información detallada proporcionando su facilidad sobre su instalación y funcionamiento, por lo que nos da una gran ventaja para crear determinadas topologías (Mininet, 2023.)

Mininet puede ser implementado en cualquier PC, como se visualiza en la Figura 14, ya que no es un software exigente en términos de recursos, y su diseño favorece un desarrollo eficiente de redes. Los diseños son versátiles y compatibles con otros programas, lo que facilita compartir configuraciones. Esta herramienta posibilita realizar pruebas de topologías complejas sin requerir una red física, permitiendo la ejecución en el código de Python (MININET, 2023).

Figura 14.
Esquema MININET

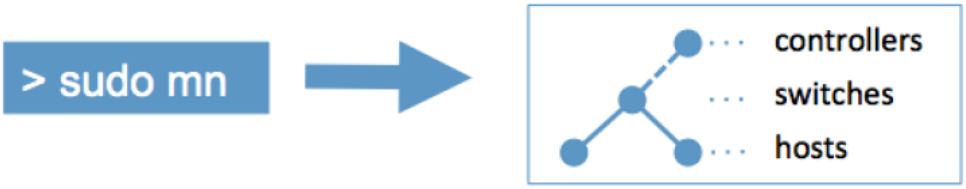


Fuente: Mininet,2023

Mininet ofrece la capacidad de emular diversos tipos de envío de paquetes mediante configuraciones de anchos de banda o Calidad de Servicio (QoS) en interfaces Ethernet reales. Además, proporciona herramientas de diagnóstico que suministran información detallada sobre el estado de la red emulada. La flexibilidad de Mininet se extiende al permitir escalar redes con miles de switches en una única máquina física.

La interfaz de línea de comandos de Mininet facilita la creación de topologías, desde las más simples hasta las verdaderamente complejas, con un solo comando. Este comando, representado en figura 15 ejecuta el kernel real, el conmutador y el código de la aplicación, brindando así una solución integral para la emulación de redes (Mininet, 2023).

Figura 15.
Mininet / Estructura de Comando



Fuente: Mininet, 2023

Su comando principal: «**sudo mn**», constituye el punto de inicio para generar una red y presenta múltiples opciones para su personalización. Se pueden añadir especificaciones adicionales a este comando para crear una red SDN de manera más personalizada y adaptada a los requisitos particulares.

En la Tabla 3 se mencionan los comandos más comunes y utilizados:

Tabla 3.
Comandos «sudo mn»

Root	Mininet	Guión	Opción	Parámetro	Argumentos
			help		Visualizar los comandos disponibles.
			node		Dispositivos de red desplegados.
			dump		Información sobre los dispositivos desplegados.
			net		Muestra la interconexión de los dispositivos.
			switch	Ovs	Trabaja con Open vSwitch.
			controller	Default	Trabaja por defecto con el controlador instalado en la máquina virtual. Se especifica su dirección Ip.
sudo	mn	--			

	topo	Mínimal/ single/ linear	**Mininal**, crea una topología por defecto 1 switch y 2 host. **Single**, 1 solo switch conectado a n host. **Linear,** crea n switches, conectado uno detrás de otro, cada uno con 1 host conectado.
	mac	Dirección ip	Asigna direcciones MAC fáciles al host de manera automática.

Fuente: Elaboración propia

Al ingresar a nuestro CLI de Mininet, se considera los siguientes comandos presentados en la tabla 3:

Tabla 4.

Comandos CLI de Mininet

Comando	**Descripción**
Exit	Finaliza la emulación
Help	Muestra información
Dump	Información detallada de la red
Net	Información de enlaces
Nodes	Listado de nodos usados
Links	Reporte de enlaces operativos
Pingall	Prueba la conexión de la red
Iperf	Rendimiento ancho de banda TCP
Iperudp	Rendimiento ancho de banda UDP
Xterm	Abre consolas independientes

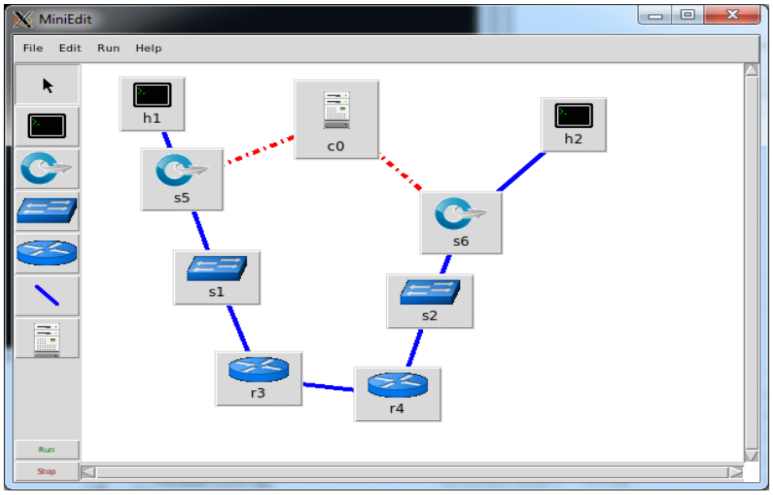
Fuente: Elaboración Propia

Mininet cuenta con una extensión llamada MiniEdit, que se presenta como una alternativa para la creación de topologías de red mediante un entorno gráfico. Esta extensión ofrece una interfaz de usuario sencilla, que presenta una pantalla con una línea de iconos de herramientas en el lado izquierdo de la ventana y una barra de menú en la parte superior, como se visualiza en la Figura 16. Este enfoque gráfico facilita la configuración y visualización de topologías de red de manera más intuitiva y accesible (MiniEdit , s.f.).

Una extensión altamente beneficiosa de Mininet para la creación de topologías personalizadas y complejas son aquellas basadas en scripts de Python. Estas representan una herramienta idónea para diseñar topologías a medida según nuestros requisitos específicos. Es esencial crear o modificar un script en Python, el cual se encuentra

ubicado en la carpeta de descarga de Mininet, específicamente en el directorio "custom" (MININET, s.f.).

Figura 16.
MiniEdit



Fuente:MiniEdit, s.f.

- **EstiNet**

Esta herramienta es un emulador y simulador de redes SDN, con soporte de OpenFlow, cuenta con un entorno gráfico, lo que nos permite visualizar el comportamiento de la red establecida. (Planas, 2016). EstiNet no es un software libre, por lo que al ser un producto comercial no cuenta con demasiada información, y además cabe mencionar que la herramienta corre sobre un SO Fedora (Lasso Guamàn & Puchaicela, 2021).

EstiNet se configura como una herramienta de software integral que abarca la planificación, pruebas, educación, desarrollo de protocolos y la predicción de rendimiento de aplicaciones. Desarrollado en lenguaje Python, lo que nos permite crear topologías personalizadas, con un número alto de nodos.(EstiNet, s.f)

Para llevar a cabo estas funciones, EstiNet adopta la metodología de «kernel re-entrada», que se ejecuta a través de una red de túneles. Específicamente, utiliza interfaces para capturar paquetes enviados desde la capa IP en el kernel de Linux y los dirige hacia el motor de simulación. Esta metodología optimiza la implementación de EstiNet, asegurando una ejecución eficiente de sus diversas funcionalidades (Politécnica, n.d.).

- **NS-3**

Se trata de un simulador de eventos discretos, concebido principalmente para fines educativos e investigativos, que se inició como un proyecto de código abierto en 2006. Aunque ha sido diseñado con la intención de facilitar la simulación en entornos educativos y de investigación, la disponibilidad limitada de información puede dificultar la obtención de detalles necesarios para llevar a cabo simulaciones de trabajo de manera efectiva.

Simulador de redes, posee diferentes librerías que le dan soporte para poder simular dispositivos OpenFlow, es mucho más escalable y realista que otros simuladores, y cabe mencionar que, aunque es de código libre no permite la integración de controladores SDN externos (Pacuar, 2022).

NS-3, ha sido desarrollado para proporcionar una plataforma de simulación de red abierta y extensible para la investigación y la educación en redes, es decir proporciona modelo de cómo funcionan las redes de datos por paquetes y proporciona un motor de simulación para que los usuarios realicen experimentos de simulación. Mientras que algunas plataformas de simulación proporcionan a los usuarios un entorno de interfaz gráfica de usuario único e integrado en el que se llevan a cabo las tareas, NS-3 es más modular en este sentido.

NS-3 se utiliza principalmente en sistemas Linux o MacOS, aunque tiene soporte para algunos otras más modelos de simulación. El proyecto se compromete a construir un núcleo de simulación sólido que esté bien documentado, sea fácil de usar y depurar, y que satisfaga las necesidades de todo el flujo de trabajo de simulación, desde la configuración de la simulación hasta la recopilación y el análisis de seguimiento.

Además, la infraestructura del software NS-3 fomenta el desarrollo de modelos de simulación que sean lo suficientemente realistas como para permitir que se utilice como un emulador de red en tiempo real, interconectado con el mundo real, y que permite que muchas implementaciones de protocolos existentes en el mundo real sean reutilizadas dentro de NS-3.

El núcleo de simulación NS-3 admite la investigación en redes IP y no basadas en IP. Sin embargo, la gran mayoría de sus usuarios se centra en simulaciones inalámbricas/IP que involucran modelos para Wi-Fi, LTE u otros sistemas inalámbricos

para las capas 1 y 2. Otros temas de investigación populares incluyen el rendimiento de TCP y el rendimiento del protocolo de enrutamiento móvil ad hoc.

NS-3 también admite un programador en tiempo real que facilita una serie de casos de uso de «simulación en el bucle» para interactuar con sistemas reales. Por ejemplo, los usuarios pueden emitir y recibir paquetes generados por NS-3 en dispositivos de red reales, y NS-3 puede servir como marco de interconexión para agregar efectos de enlace entre máquinas virtuales.

Otro énfasis del simulador está en la reutilización de aplicaciones reales y código del kernel. El marco de ejecución directa de código permite a los usuarios ejecutar aplicaciones basadas en C o C++ o la pila de redes del kernel de Linux dentro de NS-3 (Ns-3, s.f).

- **GNS-3**

Simulador de red escalable, con un entorno grafico amigable, que nos permite visualizar las distintas configuraciones de la arquitectura de red, puesto que ya no se tiene la necesidad de la manipulación de hardware de red, es de código libre, y permite la integración de máquinas virtuales. (Jácome & Naranjo, 2022)

Definido y establecido por GNS3, se presentan sus mejores características, porque es un simulador de software libre que brinda la capacidad de diseñar topologías de redes complejas y extensas. Este simulador no solo facilita la creación de entornos de red detallados, sino que también permite la interoperabilidad con máquinas virtuales mediante virtualizadores como VirtualBox. Una característica destacada de GNS3 es su capacidad para emplear los sistemas operativos internos (IOS) de los equipos de Cisco y ejecutarlos en un entorno virtual en el ordenador, simulación representada en la Figura 17.

La emulación que ofrece GNS3 puede llevarse a cabo en computadoras basadas en sistemas operativos Windows, Linux y Mac OS.

Características de GNS3

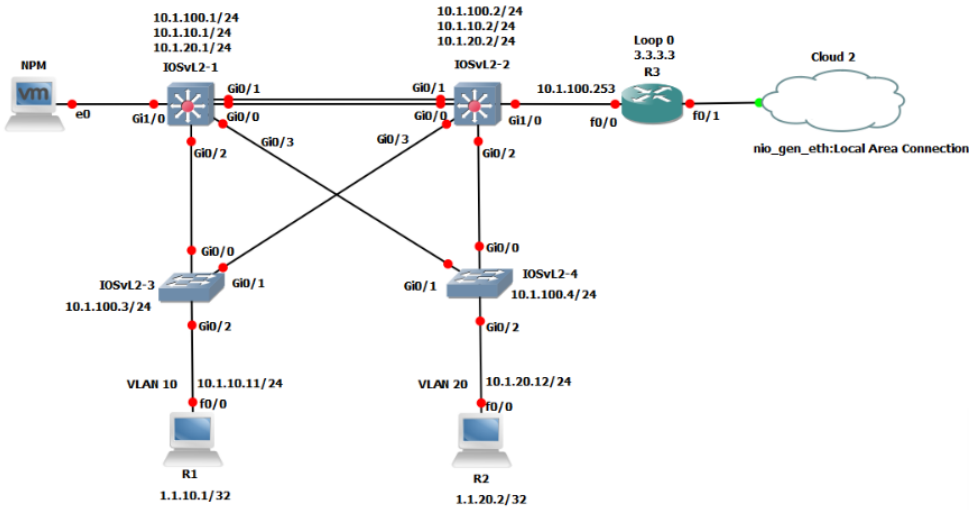
Las principales características que tiene el emulador GNS3 son las siguientes:

- Diseño de topologías de redes de alta calidad y complejidad.
- Emulación de muchas plataformas de IOS de routers Cisco IOS, switch firewalls y otros.
- Admite múltiples opciones de conmutación

- Conexión de redes simuladas al mundo real.
- Captura de paquetes utilizando Wireshark.
- Dispositivos descargables, gratuitos, preconfigurados y optimizados disponibles para simplificar la implementación.
- Se encuentra en un proceso constante actualización, con versiones de la aplicación periódicamente.

Figura 17.

Simulación de Topologías



Fuente: GNS-3, s.f.

Es relevante destacar que GNS3 es un programa de código abierto y de uso gratuito. Su capacidad para interactuar directamente con los sistemas operativos internos (IOS) de Cisco resalta como una característica poderosa. Esta integración permite observar el comportamiento real de los equipos y probar configuraciones utilizando imágenes auténticas de los dispositivos. Además, GNS3 ofrece la opción de conectar las topologías creadas en el simulador con hardware real, ofreciendo así una flexibilidad adicional para integrar entornos virtuales y físicos (GNS3, s.f.).

5. Metodología

5.1. Área de Estudio

La presente investigación se desarrolló en el campus principal de la Universidad Nacional de Loja, ubicado en el sector la Argelia. La UNL cuenta con 5 facultades las cuales se mencionan a continuación:

- Facultad Agropecuaria y de Recursos Naturales Renovables
- Facultad Educación, el Arte y la Comunicación
- Facultad Energía, las industrias y los Recursos Naturales no Renovables
- Facultad Jurídica, Social y Administrativa.
- Facultad de Salud Humana

Además, cuenta con dependencias como:

- Modalidad de Estudios a Distancia (MED)
- Administración Central
- Campus Universitario Motupe

De las dependencias mencionadas dos son externas: la Facultad de Salud Humana y el Campus Universitario Motupe, por lo cual no se consideran en el diseño de la propuesta basada en SDN para la red de datos.

5.2. Tipo de investigación

En este proyecto se realizó una investigación bibliográfica y experimental para recopilar información acerca de las SDN, funcionamiento, herramientas, y tipos de mejoramiento que pueden implementarse en las redes, se creó y simuló las topologías de la red de datos del campus La Argelia, obteniendo información relevante sobre el rendimiento de la red.

- **Métodos de investigación**

Se utiliza los siguientes métodos para realizar la investigación:

- ✓ **Método analítico**

Es una técnica de investigación que se caracteriza por descomponer un fenómeno complejo en partes simples y manejables, con el objetivo de comprender su funcionamiento. En esta investigación se realizó un análisis del rendimiento de las redes SDN en comparación con la red tradicional de la UNL, en las que se mencionan las ciencias exactas y naturales que se complementan con análisis discursivos para cualificar y dar precisión formal a los resultados obtenidos. (Lopera et al., 2010)

- ✓ **Método comparativo**

El método comparativo se basa en la lógica inductiva y se utiliza para estudiar fenómenos a través de diferentes situaciones con el objetivo de identificar variables y factores básicos y constantes. En el contexto de este proyecto, se trata de una comparación de funcionamiento y rendimiento de una red tradicional y una solución propuesta con SDN, con el objetivo de identificar las ventajas que ofrece la nueva tecnología. Este enfoque le permite analizar exhaustivamente las diferencias y similitudes entre sistemas, identificar áreas de mejora y evaluar el impacto de la implementación de tecnologías innovadoras en la red (Lopera et al., 2010).

- ✓ **Método científico**

Este método permite registrar acontecimientos que preceden a la realización de la investigación, tal como puede representarse formalizando y sometiendo a la experiencia formulaciones teóricas con el propósito permanente de comprobar su validez y tratar de modificar la realidad a la que se adaptan (Lopera et al., 2010).

5.3. Selección de Herramientas

Para la selección de las herramientas adecuadas, se llevó a cabo una comparación entre las distintas opciones presentadas, de los simuladores, emuladores, protocolos y controladores, los cuales apoyan la investigación en el campo de las SDN. Se establecen sus características principales para mejorar la experimentación realista de las SDN. La cuantificación exacta de cada característica dependerá del modelo implementado en cada herramienta:

- ➢ **Herramienta Software de simulación y emulación:**

Es importante saber que existe diferencia entre un simulador y un emulador de redes, el emulador es una herramienta que ejecuta una copia exacta del sistema operativo de red (generalmente consumen mayor cantidad de recursos del equipo en que se instala), mientras que un simulador está diseñado para tener una semejanza con un sistema operativo de red. Considerando estos aspectos resulta un desafío para estas herramientas el análisis de resultados de rendimiento obtenidos mediante la transmisión de datos.

La numeración asignada corresponde a los niveles de importancia, donde 1 representa Baja, 2 indica Media, y 3 denota Alta. La herramienta más adecuada para el proyecto será aquella que logre la mayor sumatoria, la cual resulta de la evaluación según diversos criterios establecidos.

Métricas de evaluación para la selección del Emulador y Simulador

- ✓ **Documentación:** contar con documentación detallada, exhaustiva y actualizada fáciles de entender, con una ubicación centralizada que cubra todos los elementos claves y escenarios prácticos que ayuden a comprender la herramienta.
- ✓ **APIs:** la posibilidad de conectar sistemas mediante interfaces de programación definidas.
- ✓ **Escalabilidad:** permite que el entorno de creación de redes se pueda escalar a redes con cientos de switch en una sola computadora, dependiendo claro de los recursos de hardware.
- ✓ **Simulaciones compartibles:** permita que las simulaciones se puedan adaptar fácilmente con otros colaboradores para su ejecución.
- ✓ **Instalación**: que sea fácil de realizar, y permita poder optar por distintos procedimientos.
- ✓ **Software Libre:** que cuyo estudio y uso, cuente con la libertad de usar el software para cualquier propósito adaptándolo a nuestras necesidades.
- ✓ **Protocolos:** admita distintos protocolos para la comunicación entre las distintas capas de arquitectura, especialmente protocolo OpenFlow.
- ✓ **Plataformas (Sistemas operativos):** que cuente con un SO especializado y con el propósito específico para el uso de las distintas herramientas.
- ✓ **Adaptabilidad:** capacidad para manejar distintas características y especificaciones de equipos, realizar modificaciones sin perder funcionalidad.
- ✓ **Interfaz amigable:** fácil de entender y utilizar, para una óptima interacción entre el usuario y el software, constando de simplicidad y flexibilidad, para adaptar la interfaz según las preferencias.
- ✓ **Compatible con controladores reales:** se puede simular a un nivel de virtualización en la web, en tiempo real.

❖ **Emuladores**

Tabla 5.
Métricas de evaluación del Emulador

	Emuladores			
Características	**Mininet**	**Calif**	**Estinet**	**Calif**
Documentación Existente	Muy Buena	3	Media	2
APIs(NBI/SBI) Existentes	Muy Buena	3	Media	2

Características	Mininet	Calif	Estinet	Calif
Escalabilidad:	Media (múltiples procesos)	2	Alta (un solo proceso)	2
Simulaciones compartibles:	Alta	3	Alta	3
Instalación	Fácil	3	Fácil	3
Software libre	Si	3	No	1
Protocolos	Openflow	3	Openflow	3
Plataformas (SO)	Linux	3	Fedora	2
Total:		**23**		**18**

Fuente: Elaboración Propia

En la Tabla 5, se llevó a cabo una exhaustiva comparación entre dos emuladores, concluyendo que Mininet es la opción óptima para el proyecto de titulación. Esta elección se fundamenta en sus diversas características, destacando como factor primordial su condición de software libre. Además, Mininet se respalda en su capacidad para ofrecer ventajas significativas y proporcionar múltiples soluciones, abundante documentación, uso de diversas APIs entre otros, lo que garantiza el desarrollo óptimo del proyecto.

- **Simuladores**

Tabla 6.
Métricas de evaluación del simulador

	Simuladores			
Características	**NS-3**	**Calif**	**GNS-3**	**Calif**
Documentación Existente	Media	2	Alta	3
Adaptabilidad	Media	2	Alta	3
Escalabilidad:	Media	2	Alta	3
Simulaciones compartibles:	Baja	1	Alta	3
Interfaz amigable	Media	2	Alta	3
Software libre	Si	3	Si	3
Modo simulación:	Si	3	Si	3
Plataformas (SO)	Mac Linux	2	Mac Linux Windows	3
Protocolo	OpenFlow	3	OpenFlow	3
Compatible con controladores reales	Si	3	Si	3
Total		**23**		**30**

Fuente: Elaboración Propia

La elección de un simulador apropiado y que satisfaga las diversas características se justifica en la Tabla 6. Este aspecto adquiere relevancia, ya que se analizarán detalladamente las ventajas más significativas del software con el objetivo de presentar una simulación ideal. Es esencial mantener como una característica fundamental la capacidad de realizar simulaciones compartibles, ser utilizable en distintos sistemas operativos, ofrecer escalabilidad y, sobre todo, ser un software de código abierto. GNS3 destaca en las demás características que lo posicionan como la opción ideal para el proyecto.

- **Selección de controlador SDN**

Se definen los parámetros a tomar en cuenta.

- ✓ **Escalabilidad:** Se toma en consideración el número de conmutadores que un controlador puede manejar eficazmente, lo que conlleva su capacidad para mitigar el impacto en la congestión de la red. Entonces, la escalabilidad es fundamental para garantizar un rendimiento óptimo de los dispositivos en el plano de datos.
- ✓ **Interoperabilidad con redes tradicionales:** Permitir garantizar la comunicación efectiva entre componentes tradicionales y SDN.
- ✓ **Modularidad:** Permite al controlador la flexibilidad, mantenimiento y escalabilidad del sistema.
- ✓ **Documentación:** Contar con documentación detallada, exhaustiva y actualizada fáciles de entender, además contar una ubicación centralizada que cubra todos los elementos claves y escenarios prácticos que ayuden a comprender la herramienta.
- ✓ **Rendimiento:** En relación al tiempo requerido para construir la tabla de flujo, la cual registra las acciones que deben ser aplicadas al tráfico que llega al switch, se contempla que, si un paquete recibido no coincide con ninguna de las entradas existentes, será enviado al controlador para determinar la forma en que debe ser conmutado; en ausencia de una acción definida, el paquete será descartado. Además, se toma en cuenta la tasa de establecimiento de flujos por segundo que el controlador es capaz de manejar.
- ✓ **Programación de la red**: Al contar con interfaces para la programación del controlador, proporciona a estas varias funcionalidades.

- ✓ **Soporte de plataforma (Sistemas Operativos):** Los controladores corren sobre sistemas operativos, consecuentemente este debe ser multiplataforma permitiendo una mayor flexibilidad e independencia al momento de su implementación.
- ✓ **Procesamiento:** Verificar si puede soportar procesos múltiples o no, pues esto repercutirá en la escalabilidad de los núcleos de la CPU
- ✓ **Lenguaje:** Elegir el lenguaje adecuado puede afectar al desempeño y rapidez con el que se desenvuelve el controlador. Algunos leguajes como Java, C ++ y Python son los que utilizan con mayor frecuencia los desarrolladores para controladores de redes definidas por software.

  Cada uno de estos lenguajes presentan ventajas e inconvenientes, un controlador programado en Java destaca al ser multiplataforma y tener la capacidad de subdividir aplicaciones en pequeñas partes.

  Por otro lado, los controladores creados con lenguaje C++ brindan mayor desempeño, pero no poseen una administración de memoria buena ni una GUI amigable con el usuario.

  Los controladores programados en Python no pueden sobrellevar diversos subprocesos a la vez que permita multiplataforma y que sean de alta modularidad.
- ✓ **Software Libre:** Software de código abierto para adaptarlo a nuestras necesidades.
- ✓ **Protocolos:** Admita distintos protocolos para la comunicación entre las distintas capas de arquitectura, especialmente protocolo OpenFlow.
- ✓ **Virtualización:** Permite a los administradores de la red crear dinámicamente las redes virtuales basados en políticas, para satisfacer una amplia gama de requerimientos.

Tabla 7.
Métricas de Evaluación controlador

	Características							
Controladores	NOX	Calificación	ONOS	Calificación	Ryu	Calificación	Opendayligth	Calificación
Interoperabilidad con redes	No	1	No	1	No	1	Si	3

tradicionales								
Modularidad	Baja	1	Alta	3	Moderada	2	Alta	3
Documentación	Baja	1	Alta	3	Media	2	Alta	3
Rendimiento	Bajo	1	Media	2	Media	2	Alto	3
Programación de la red	C++	1	Web	3	Python	2	Web	3
Soporte de plataforma (SO)	Linux	1	Windows Linux macOS	3	Linux macOS	2	Windows Linux MacOS	3
Multiproceso	Si	3	Si	3	Si	3	Si	3
Software abierto:	Si	3	Si	3	Si	3	Si	3
Escalabilidad	Baja	1	Alta	3	Media	2	Alta	3
Protocolos	OF v1.0	1	Openflow 1.0-1.5	3	Openflow1.0-1.3	2	Openflow1.0-1.5	3
Virtualización	Mini-net OpenVswitch	3	Mininet OpenVswitch	3	Mininet OpenVswitch	3	Mininet OpenVswitch	3
Lenguaje de programación	C++	1	Java	3	Python	2	Java	3
Total:		**18**		**33**		**24**		**36**

Fuente: Elaboración Propia

La elección del núcleo de gestión de la red se rige como un factor determinante para establecer el éxito de la infraestructura de datos. En este sentido, en la Tabla 7 se han considerado diversas características esenciales para el controlador seleccionado entre los cuatro evaluados, la decisión final se inclinó hacia Opendayligth, destacándose de manera significativa con una ponderación superior. Aunque todos los controladores analizados son de código abierto, Opendayligth resalta especialmente en áreas clave, como interoperabilidad con redes tradicionales y rendimiento.

- **Selección de Protocolo SDN**

Protocolo estandarizado que permita la interoperabilidad de múltiples proveedores permitiendo su elección para reducir costos.

- ✓ **Compatibilidad con controladores:** Permitir la comunicación eficiente entre distintos controladores, no se limita a uno específico.
- ✓ **Soporte Mininet:** Que admita el protocolo, para el desarrollo y pruebas de las soluciones en el entorno de emulación.
- ✓ **Flexibilidad:** Capacidad de cambiar y adaptarse fácilmente en respuesta a nuevas necesidades, requisitos o condiciones.

Tabla 8.
Métricas de Evaluación protocolo

	Protocolo							
Características	NetConf	Calif	OvSDB	Calif	ForCES	Calif	OpenFlow	Calif
Controlador compatible	Opendaylight Ryu Onos	3	Opendaylight Ryu Onos	3	Especifico	1	Opendaylight Ryu Onos	3
Soporte mininet	No	1	Si	3	No	1	Si	3
Flexibilidad	Alta	3	Media	2	Alta	3	Alta	3
Total:		**7**		**8**		**5**		**9**
Nota: calif=calificación								

Fuente: Elaboración Propia

El protocolo de comunicación destacado, según los resultados de la Tabla 8, es el protocolo OpenFlow. Esta elección se fundamenta en su notable compatibilidad con diversos protocolos, su respaldo para Mininet y su flexibilidad ante los diversos requisitos y condiciones de la red. Por tanto, se posiciona como la opción más idónea para ser implementada en el proyecto.

5.4. Selección de Metodología de Diseño de Red.

Se ha considerado realizar una comparativa entre las metodologías más importantes para el diseño óptimo de una red. Entre las metodologías destacadas se encuentran las siguientes:

- **Metodología Top-Down Network Design**

Según (Guevara & Quizhpi, 2017), respaldándose en los principios establecidos por Oppenheimer en "Top-Down Network Design Third Edition" (2011), esta metodología se fundamenta en el modelo OSI al desarrollar un diseño que abarca desde las capas más elevadas hasta las inferiores. Inicia con las capas de aplicación, presentación, sesión y transporte, para luego dar prioridad a las capas de red, enlace de datos y física.

Este enfoque se adapta exitosamente a nuestro diseño de red, ya que implica un análisis exhaustivo de la situación actual de la red, los requisitos, las limitaciones y la estructura lógica que debe considerarse durante el desarrollo de la metodología.

La metodología sigue un proceso estructurado en cuatro fases, delineando los pasos a seguir:

- Fase I: Análisis de requerimientos

  Esta fase se enfoca en el análisis de la red, se recomienda describir la estructura tanto a nivel lógico como nivel físico, lo que nos permite poder obtener un panorama amplio sobre la red.
- Fase II: Diseño lógico de la red

  Diseñamos la topología de red que se quiere implementar u optimizar, seleccionando protocolos, softwares, entre otras herramientas adecuadas para mantener el mecanismo de gestión y mantenimiento de la red.
- Fase III: Diseño físico de la red

  Selección de equipos y tecnología que puede desplegarse en el diseño lógico propuesto.
- Fase IV: Probar, optimizar y documentar el diseño de la red

  Poner a prueba la red mediante simulaciones, con el fin de monitorear su rendimiento, obteniendo resultados para destacar estrategias de optimización. Se pretende documentar todo el proceso realizado del diseño de red.

- **Metodología Cisco PPDIOO.**

Las redes crecen de manera organizada, siendo necesario planificar todo desde el principio, Maggio, (2018), menciona que Cisco PPDIOO, presenta el siguiente ciclo de vida de una red:

- Preparar: Evaluar inicialmente las necesidades presentadas y la definición de una arquitectura conceptual.
- Planificar: Traducir las necesidades presentadas para poder optar por soluciones.
- Diseño: Con los conocimientos adquiridos sobre la red, se establece un diseño con la solución adecuada para la optimización.
- Implementar: Una vez aprobado el diseño, se procede a implementar instalando y configurando dispositivos.
- Operar: se procede a monitorear los distintos componentes de la red, manteniendo el desempeño, actualizaciones e identificando problemas que pueden surgir.
- Optimizar: Después de que la red está operando correctamente, si se presentan algunos problemas, se podrá identificarlos y solucionarlos de una manera óptima.

- **Metodología propuesta por James Mccabe**

Una metodología propuesta por James McCABE en su libro «Practical Computer Network Analysis and Design», (Mc Cabe & James D, 1998),divide el diseño de red en fases con el fin de mantener cambios sin dañar la estructura, las cuales son:

- Fase de Análisis. - Normalmente se detallan las redes a nivel de campus, en la que se caracterizan los flujos de información simples o compuestos, es decir, entre el origen y destino, la capacidad de soporte, entre otras características referenciales del sistema de red.
- Fase de Diseño. - Se establecen dos procesos principales que son: fase de diseño lógico, en la que consta de evaluación de tecnologías, selección de interconexión, análisis de riesgos entre otras características; y el diseño físico, evaluar el proceso para una futura implementación, teniendo en cuenta una estrategia detallada sobre todos los procesos a seguir con los equipos (Guevara & Quizhpi, 2017).

- **Selección de Metodología**

Tabla 9.

Métricas de evaluación de metodología de diseño de Red

METODOLOGÍA						
Metodología	Top-Down Network	Calificación	Cisco PPDIOO	Calificación	James MaCABE	Calificación
Capacidad de adaptación	Alta	3	Media	2	Media	2

Facilidad del despliegue de red	Media	2	Alta	3	Media	2
Disponibilidad y optimización de red	Alta	3	Media	2	Media	2
Total		8		7		6

Fuente: Elaboración propia

5.5. Aplicación de Metodología TOP-DOWN NETWORK

Para el diseño y simulación de la red de datos de la Universidad Nacional de Loja, según lo indicado en la tabla 9, la metodología TOP-DOWN se destaca por contar con características superiores y fases altamente adaptables para el desarrollo optimo del proyecto, por lo que ha sido seleccionado para el diseño de la red. Como se mencionó anteriormente, esta metodología consta de cuatro fases que se ajustan progresivamente para abordar detalles y requisitos específicos de la red de datos. Cada una de estas fases nos brinda un marco estructurado para el desarrollo del diseño, incluyendo criterios y puntos.

5.5.1. FASE I: ANÁLISIS DE REQUERIMIENTOS

Lograr una mayor administración, disponibilidad y automatización, así como un rendimiento óptimo en la red, constituye uno de los objetivos centrales de este proyecto.

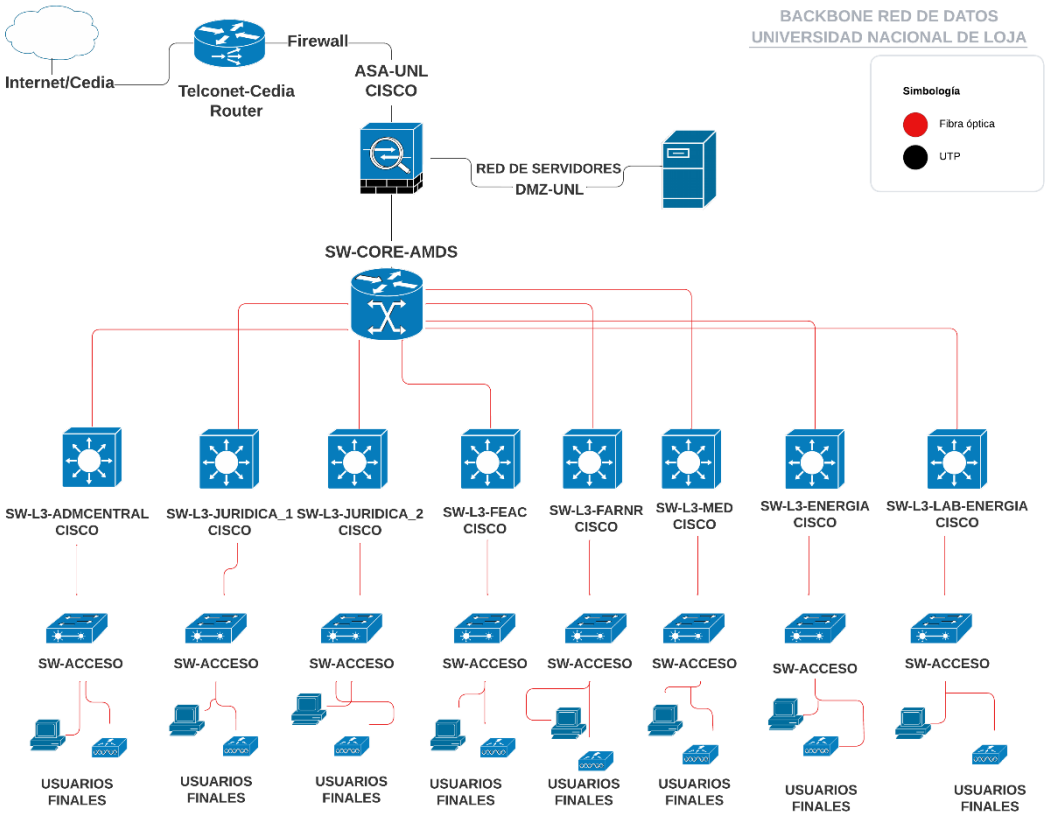
Sin embargo, se identifica un desafío primordial que requiere atención: el rendimiento de la red, afectado por variables como la latencia, el jitter y pérdida de paquetes. Es importante destacar que la red implica múltiples saltos antes de llegar al usuario final, generando inevitablemente cierto retardo en la transmisión de datos.

- **Diseño de la red Actual**

Enfocándonos en llevar a cabo la simulación de red mediante el uso del software GNS3, el cual posibilita la representación virtual de la red, se busca obtener información detallada acerca del funcionamiento y rendimiento de la misma.

La dependencia encargada de la operación y mantenimiento de la red de datos de la UNL, es la **Unidad de Telecomunicaciones e Información (UTI)**. Esta unidad es la que alberga los equipos principales de la red de datos.

Figura 18.
BACKBONE RED DE DATOS UNL



Fuente: Elaboración Propia

La conexión de la red se realiza a través de fibra óptica con el proveedor de servicios de internet llamado Consorcio Nacional para el Desarrollo de Internet Avanzado (CEDIA), que opera bajo la red Nacional TELCONET. Este servicio brinda un ancho de banda establecido de 2.4 Ghz, en el que se utiliza un 60% (según la información brindada por parte de subdirección de redes - UTI). El monitoreo de la red de datos se realiza mediante Zabbix, software libre utilizado para la red. Su arquitectura cuenta con 3 capas o niveles de servicio los cuales son: **núcleo, distribución y acceso**.

La red de datos de la UNL consta de distintos equipos interconectados para ofrecerles los distintos servicios a los usuarios finales, desde un wireless LAN controller para los distintos puntos de acceso inalámbrico de la universidad, que están conectados a un switch Cisco 2960, de capa 2, interconectado hasta un switch Cisco WS-C3750, de capa 3, los mismos que se encuentran ubicados en cada una de las facultades de la UNL.

La red abarca un extenso territorio con numerosos puntos de acceso y switches, lo que provoca que la gestión se torne complicada debido a que muchos de los dispositivos

requieren configuraciones individuales. Esto dificulta la labor del administrador, por lo que se sugiere un cambio de tecnología en la administración de las redes.

Es esencial obtener una comprensión precisa de las disparidades entre las infraestructuras de SDN y las redes tradicionales de manera general. Este conocimiento resulta crucial ya que nos permite identificar con claridad las ventajas inherentes a la implementación de un diseño basado en tecnologías innovadoras. La información detallada, expuesta en la Tabla 10, específica y resalta los puntos clave que respaldarán nuestro enfoque hacia una propuesta de diseño actual y mejorada.

Tabla 10.
Comparativa Redes Tradicional/SDN

RED TRADICIONAL	**Propuesta RED SDN**
Arquitectura estática y compleja	Directamente dinámica
Configuración descentralizada	Inteligencia centralizada
Falta de escalabilidad	Escalable
Políticas inconsistentes para cada equipo	Tablas de enrutamiento
Dependencia de proveedores	Estándares abiertos
Altos costos de recursos financieros	Facilidad de innovación

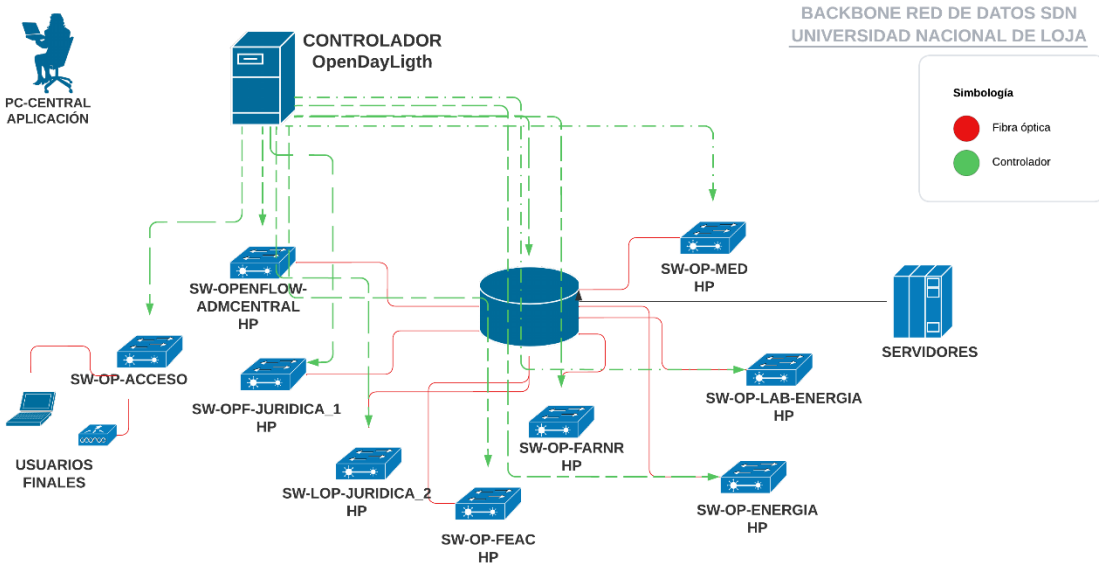
Fuente: Elaboración Propia

5.5.2. FASE II: DISEÑO LOGICO DE LA RED

- **Topología de SDN**

En esta fase, detallamos la ejecución de la simulación de una red SDN, la cual fue desarrollada con la similitud posible respecto a la red de datos actual, con el fin de llevar a cabo una comparativa más objetiva.

Figura 19.
Diseño red SDN



Fuente: Elaboración Propia

El diseño lógico de la red tuvo en consideración todos los problemas identificados, permitiendo así la implementación de esquemas que ofrecen soluciones mediante una concentración lógica en lugar de física. Cada facultad o área administrativa cuenta con usuarios finales, y la fibra óptica se establece como el medio de transmisión principal para la red de datos en las distintas facultades del campus La Argelia.

La topología simulada sigue la infraestructura de la red de datos de la UNL, campus La Argelia, la cual se compone de las siguientes secciones:

- **Switch de Core:** Este constituye el núcleo central de comunicaciones en la red, sirviendo como el punto central desde el cual se establecen los enlaces hacia los nodos de distribución.
- **Switches de Distribución:** Estos switches se conectan directamente al Switch Core y actúan como puntos intermedios entre este y los nodos de acceso. Su función principal es dirigir eficientemente el tráfico hacia los diferentes segmentos de la red.
- **Acceso:** Los nodos de acceso son los siguientes en la jerarquía y están conectados a los interruptores de distribución. Su función es proporcionar la conectividad directa a los usuarios finales y a los dispositivos de red locales.

- **Usuarios Finales**: Estos constituyen los extremos de la red, representando los puntos de conexión directa para los usuarios y dispositivos dentro del campus. En esta configuración, los enlaces se extienden desde el nodo central de comunicaciones hacia los nodos de distribución, y posteriormente, desde estos nodos hacia los nodos de acceso. Este diseño jerárquico busca optimizar la eficiencia y la gestión del tráfico en la red, siguiendo la estructura tradicional de la red de datos.

- **Recursos**

Para demostrar la funcionalidad de la tecnología SDN y satisfacer los requisitos establecidos, se dispuso de los siguientes recursos. Las simulaciones se llevaron a cabo en una computadora portátil HP equipada con un procesador Intel(R) Core (TM) i7-1065G7 CPU @ 1.30GHz 1.50 GHz y 12 GB de memoria RAM.

Se empleó el hipervisor Oracle VirtualBox para crear la máquina virtual, realizando configuraciones específicas antes de establecer la red. Este proceso incluyó la instalación y configuración del sistema operativo Ubuntu versión 20.04, en esta máquina virtual, posteriormente, se instaló Mininet en su versión 2.3.1b4 y el controlador OpenDayLight en su versión Lithium.

- **Software**

El diseño de la red está compuesto por dos softwares principales: Mininet y controlador OpenDayLight (ODL), por lo que la elección del sistema operativo Ubuntu se fundamentó en la compatibilidad de Mininet, que está basado en este sistema operativo, aprovechando su kernel. Asimismo, esta elección se ampliará al software OpenDayLight, asegurando una integración efectiva con el entorno de simulación, los cuales se escogieron por licenciamiento gratuito, compatibilidad con Openflow, lenguaje de programación entre otras características detallados en la Tabla 4 y 7.

En el diseño de red SDN, se desarrolló un script en Python, extensión del software de Mininet, código que cumple con la topología de la red de datos actual de la UNL. Este script, contiene la secuencia de comandos, que simplifica la configuración y ejecución de la simulación de red. Permite crear de una manera fácil switches virtuales y host, utilizando el protocolo OpenFlow en la interfaz southbound (SBI), encargado de facilitar la comunicación entre los dispositivos de red y el controlador. Estos dispositivos están conectados al controlador ODL a través de una dirección IP especifica, mediante la cual

se gestiona el flujo de tráfico en la red utilizando una interfaz de programación de aplicaciones (API) Northbound, la cual se comunica con el controlador.

Para visualizar la red creada previamente en Mininet, se utiliza **Cisco OpenFlow Manager (OFM)**.

5.5.3. FASE III: DISEÑO FÍSICO DE LA RED

Se evaluaron diversos dispositivos con la capacidad de respaldar la tecnología SDN con el objetivo de mejorar el rendimiento de la red. Se busca una transición gradual en la infraestructura de red del campus La Argelia, basándonos en las especificaciones técnicas y la capacidad de transmisión de estos equipos. Se identifican los elementos esenciales y los cambios necesarios en la red para lograr una optimización significativa.

Esto permitirá determinar el presupuesto necesario para una posible implementación en el campus, que se detallará en el apartado 6.8.

5.5.4. FASE IV: PROBAR, OPTIMIZAR Y DOCUMENTAR EL DISEÑO DE LA RED

- **Métricas de evaluación**

La obtención de datos se la realizó en base a la observación de parámetros capturados por el sistema de monitoreo de red, garantizando una calidad de servicio optima.

- **Latencia**

Denominada como la medida de los retrasos en un sistema de red. Se expresa en milisegundos, y representa el tiempo que tardan los datos desde que se envió desde un sistema origen hasta que son recibidos por parte del sistema de destino. Aunque los datos teóricamente deberían atravesar casi a la velocidad de la luz, en la práctica, los paquetes de datos se mueven por la red a una velocidad menor debido a los diversos factores como la distancia, infraestructura y otras variables, constituyendo la suma de estos retrasos, la latencia de red (IBM, n.d.).

- **Jitter**

Jitter es una variación o demora en la entrega de paquetes de datos a través de una red, es decir, una inconsistencia en el tiempo de llegada de los paquetes, especialmente en las comunicaciones de tiempo real. El retraso/variación/cambio en el tiempo es una interrupción en la secuencia ordinaria de envío de paquetes de datos y se mide en milisegundos (ms) (IONOS, n.d.).

- **Pérdida de paquetes**

La pérdida de paquetes es el porcentaje de datos que se pierden durante la transmisión en una red. Esto ocurre cuando algunos paquetes no llegan a su destino debido a errores en la transmisión de datos o a que la red está demasiado congestionada. Por ejemplo, si de cada 100 paquetes solo llegan 91, la pérdida de paquetes sería del 9%. Es como si algunos mensajes se perdieran en el camino antes de llegar a su destino.

- **Evaluación de Desempeño**

Durante la ejecución, se establecieron solicitudes simultaneas de conexión a diferentes servidores utilizando el protocolo ICMP, conexiones UDP y TCP, desde cada uno de los hosts virtuales de la red. La elección de utilizar el comando ping para llevar a cabo estas pruebas es común, ya que proporciona una manera efectiva de medir la pérdida de paquetes, latencia y la variación en la entrega de paquetes.

El objetivo de las pruebas de desempeño es cuantificar diversas métricas, en este caso, medir la latencia, jitter y pérdida de paquetes. Estas pruebas se realizaron enviando tramas con tamaños de **256, 512, 1024 y 1518** bytes, siguiendo las indicaciones del **RFC 2544** de "Metodología de Evaluación".

Este RFC define una serie de pruebas que se utilizan para describir las características de rendimiento de un dispositivo de interconexión de red, la metodología de prueba permite definir varios parámetros, como los diferentes tamaños de trama que se examinarán, lo que nos permite certificar los parámetros de funcionamiento de la red de datos simulada.

De igual forma la implementación de la normativa ITU-T G.1010 es crucial, ya que proporciona parámetros claros y estandarizados para evaluar la calidad de servicio en redes de datos, específicamente en métricas como umbrales de tiempo. Esta normativa garantiza que los resultados obtenidos sean consistentes, medibles y comparables con otros estudios. Adherirse a la ITU-T G.1010 permitirá contrastar de manera efectiva el desempeño entre la red tradicional y la red SDN, documentación que se encuentra en el anexo 27.

Se realizaron múltiples mediciones durante la hora pico para garantizar una cantidad representativa de datos que faciliten la interpretación de los indicadores. Dado que la mayoría de los recursos están en uso durante este período, es crucial mantener una frecuencia de medición superior a 120 segundos para obtener datos significativos y relevantes.

Para obtener la información requerida, se emplearon distintos comandos. Por ejemplo, el comando **ping** se utilizó para iniciar la simulación y probar la conectividad entre todos los hosts, permitiendo así determinar la cantidad de paquetes perdidos. Asimismo, se empleó el comando **hping3**, una herramienta que permite enviar paquetes por puertos determinados para las solicitudes a los servidores locales considerados, aquí se hace uso de puertos TCP y UDP, lo que también facilita calcular el jitter.

La optimización prevista para la red de datos se documentó en el Anexo 3 de manera secuencial, detallando el proceso realizado para obtener los resultados deseados, en el contexto de la propuesta de diseño basada en SDN.

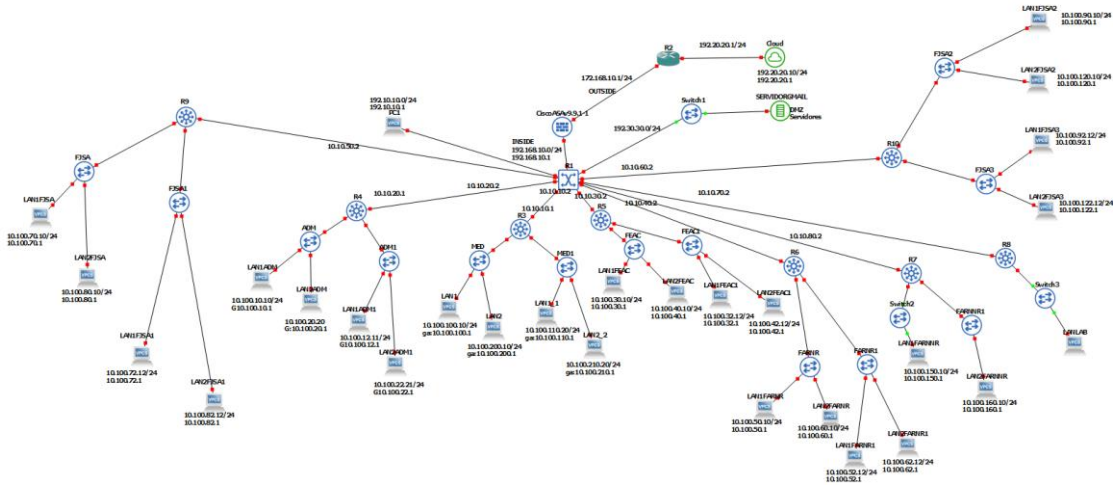
6. Resultados

6.1 Red de datos tradicional

Para procesar los datos obtenidos de la simulación en el software GNS-3, siguiendo el modelo jerárquico de tres capas (núcleo, distribución, acceso) y manteniendo la similitud con la red de datos del campus La Argelia, se utilizó una infraestructura basada en sistemas operativos IOS de CISCO, los cuales son los equipos empleados actualmente.

En esta infraestructura, representada en la Figura 20, se encuentra un firewall CISCO ASA como punto de entrada, conectado al switch Core. A partir de este switch, se establece la interconexión con los switches de distribución ubicados en cada facultad, y estos a su vez conectados a los switches de acceso. Estos switches de acceso, a su vez, están conectados a usuarios finales. Además, se ha definido una zona desmilitarizada (DMZ) donde se alojan los servidores (**Servidor DNS (port 53), Servidor web Https (port 443) y Http(port 80)**) de la intranet de la red de datos de la UNL.

Figura 20.
Simulación de Red de Datos/UNL



Fuente: Elaboración Propia

6.2 Latencia Red Tradicional

Para obtener los resultados del análisis comparativo entre el rendimiento de un entorno de red tradicional y un escenario de red creado bajo el paradigma de SDN, se realizaron conexiones a través del protocolo ICMP y conexiones hacia puertos TCP y UDP contra los servidores locales.

Los equipos desde los cuales se obtuvieron los resultados del rendimiento fueron los distintos hosts que se encuentran interconectados con los conmutadores Core,

Distribución y Acceso. Es decir que los hosts representan a los usuarios finales, que envían solicitudes de eco ICMP, hacia los servidores locales.

Basados en el RFC 2544, que considera una metodología de evaluación comparativa de los dispositivos de interconexión de red promovida por la IETF.

Se empleó la media aritmética. Este mecanismo permitió agrupar en valores promedios los resultados obtenidos, representados en el tiempo de respuesta en milisegundos, los pings de pruebas pueden ser visualizados en el anexo 2 de este documento.

En la Tabla 11 se registraron los valores obtenidos con un envió de longitud de trama de **256, 512, 1024 y 1518** bytes a los distintos servidores alojados en la DMZ, por la Línea de comandos de GNS3.

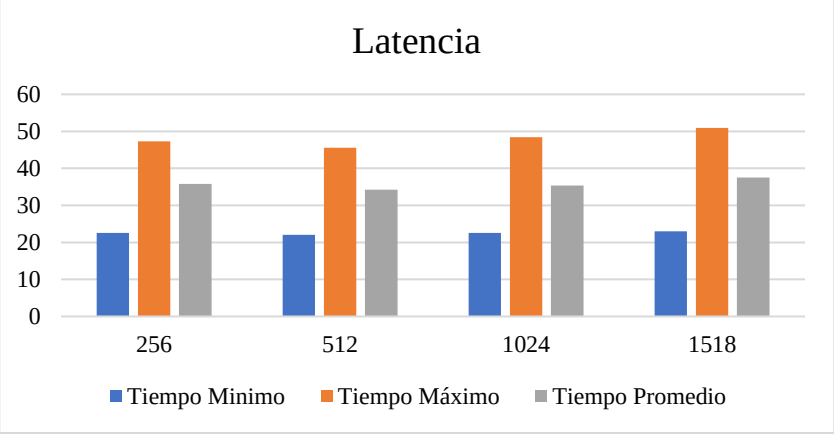
Tabla 11.
Latencia (Longitud de Paquetes)

Longitud de paquete	Tiempo mínimo (ms)	Tiempo Máximo (ms)	Latencia promedio (ms)
256 Bytes	22.59	47.33	35.81
512 Bytes	22.06	45.60	34.28
1024 Bytes	22.55	48.39	35.33
1518 Bytes	23.01	50.91	37.49

Fuente: Elaboración Propia

Para obtener una mejor perspectiva de las mediciones efectuadas y de sus resultados se procede a presentarlos de manera gráfica, como se muestra en la Figura 21, en la que se visualizan los paquetes enviados con las diferentes longitudes de paquetes (256, 512, 1024 y 1518 bytes) mismas que han utilizado un ancho de banda constante.

Figura 21
Latencia Promedio



Fuente: Elaboración Propia

6.3 Jitter en la Red Tradicional

Mediante mediciones precisas se ha analizado la variabilidad en el retardo de paquetes proporcionando una visión detallada que podrían afectar la calidad de las comunicaciones.

En el Tabla 12 se registran los datos referentes a JITTER obtenidos de las diferentes pruebas realizadas desde las PC's virtuales hacia los servidores locales simulados.

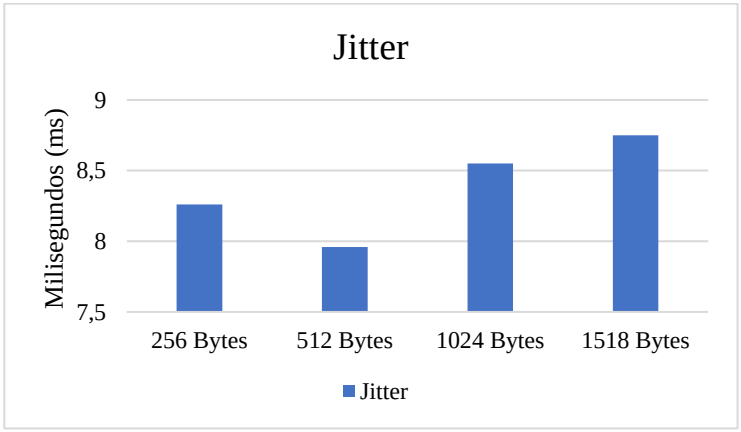
Tabla 12.
Jitter

Longitud de Paquetes	JITTER
256 bytes	8.26 ms
512 Bytes	7.96 ms
1024 Bytes	8.55 ms
1518 Bytes	8.75 ms

Fuente: Elaboración Propia

La gráfica de barras ilustra la variabilidad en los tiempos de respuesta a través de la red para cada longitud de paquete, proporcionando una representación clara y concisa de estos datos.

Figura 22.
Jitter Promedio



Fuente: Elaboración Propia

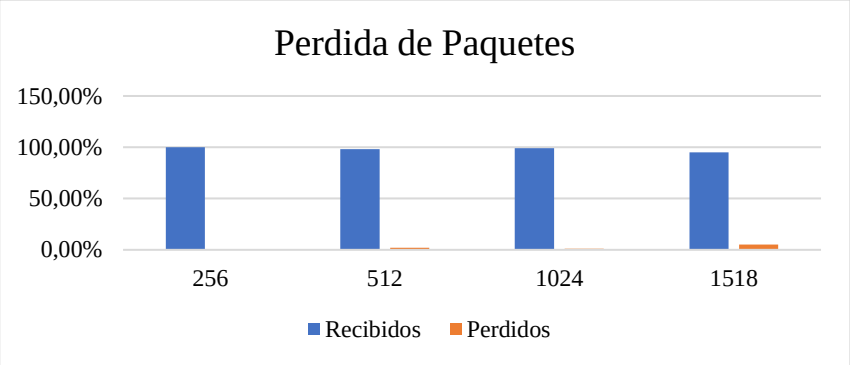
6.4 Pérdida de Paquetes

La pérdida de paquetes en una red de datos se refiere al fenómeno donde los paquetes de datos enviados desde una fuente no llegan a su destino previsto. La pérdida de paquetes se representa en la Figura 23, en la que la barra de color azul representa los paquetes enviados y recibidos exitosamente, y la barra de color anaranjado las perdidas presentadas en la trasmisión de paquetes.

En el escenario de la longitud de paquete de 256 bytes, no se han registrado pérdidas, lo que indica una conexión estable y eficiente. Sin embargo, con la longitud de paquete de 512 bytes, se ha observado una pérdida del 2% de los paquetes enviados, lo que sugiere posibles problemas de congestión o errores en la red. En el caso de la longitud de paquete de 1024 bytes, se ha observado una pérdida del 1%, lo que indica que el 99% de los paquetes fueron recibidos correctamente. Finalmente, en la longitud de trama de 1518, se ha registrado una pérdida del 5% de los paquetes, lo que sugiere que el rendimiento de la red y la entrega de datos ya presenta inconvenientes. Estos datos nos proporcionan información crucial sobre el rendimiento y eficacia de la red, y nos permiten identificar áreas que pueden requerir atención y optimización.

Figura 23.

Pérdida de Paquetes



Fuente: Elaboración Propia

6.5 Diseño Red SDN

La topología se simula en Mininet, conectado al controlador ODL, permitiendo así una visualización por parte de la GUI de la red, el script de Python utilizado se encuentra en el Anexo 2. En la **Figura 24** se observa el comando para creación de la red.

En la Tabla 13 permite presentar de manera ordenada cada aspecto relevante de la configuración y uso del entorno de Mininet, facilitando la comprensión de los componentes:

Tabla 13.

Descripción de Comandos

Comando	Descripción
sudo mn	Comando principal para el entorno de Mininet
--custom	Nombre del script de Python utilizado
--topo	Creación de la topología RedUNL, descrita en el script
--controller	Designación del controlador (ODL de manera remota)
IP	Dirección IP específica del controlador
Port	Puerto establecido para la comunicación entre Mininet y el controlador ODL

Comando	Descripción
--switch	Switch establecido para la red Open Virtual Switch
--protocols	Protocolo utilizado para las comunicaciones SBI (OpenFlow)

Fuente: Elaboración Propia

El archivo se ejecuta definiendo la personalización de la red, con sus distintos parámetros, a través del siguiente comando:

sudo mn --custom=redUNL.py --topo=RedUNL --controller=remote, ip=10.0.2.15, port=6653 –switch ovs, protocols=OpenFlow10

Figura 24.
Ejecución de Script Python

```
diana@diana-VirtualBox:~$ sudo su
[sudo] contraseña para diana:
root@diana-VirtualBox:/home/diana# cd mininet/custom
root@diana-VirtualBox:/home/diana/mininet/custom# sudo mn --custom=redUNL.py  --
topo=RedUNL --controller=remote,ip=10.0.2.15,port=6653 --switch ovs,protocols=Op
enFlow10
```

Fuente: Elaboración Propia

Al ejecutar el comando se crea la red SDN deseada:

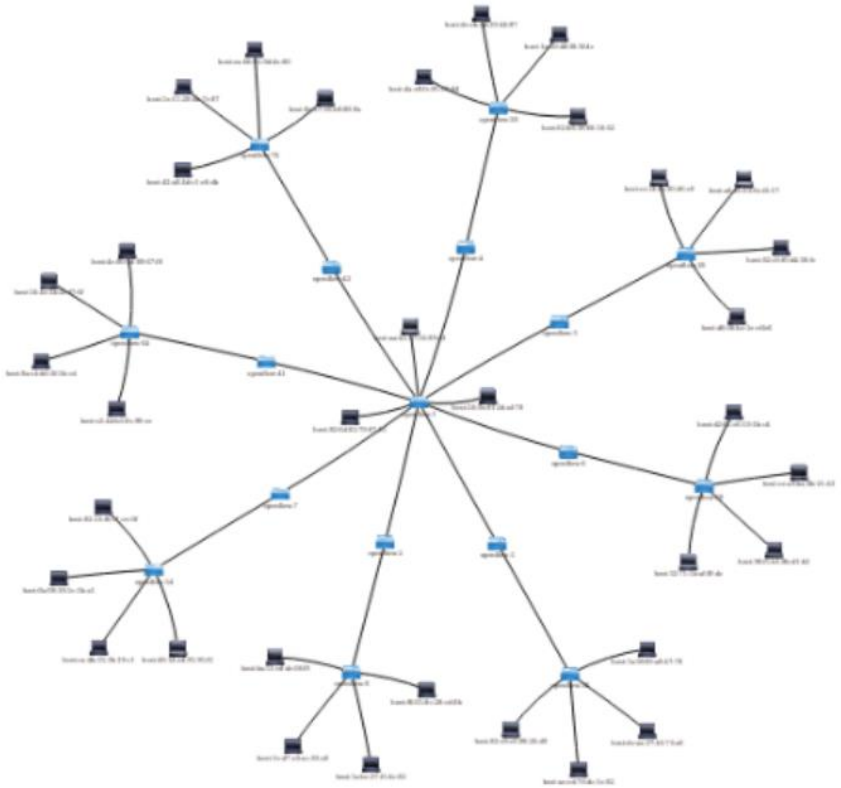
- 1 switch Core
- 8 switch de Distribución
- 8 switch de acceso
- 4 host por facultad

La topología de red presentada en la **Figura 25** en el módulo DLUX de OpenDaylight (ODL) exhibe la arquitectura completa de la red simulada, así como los hosts conectados a cada switch de acceso. Esto nos proporciona una visualización integral y práctica de la red de manera eficiente.

Cabe destacar que para que el controlador pueda identificar la ubicación de los dispositivos, es imperativo enviar tráfico a través de la red, utilizando los comandos previamente mencionados en la tabla 4.

Figura 25.

Diseño de Red basado en SDN

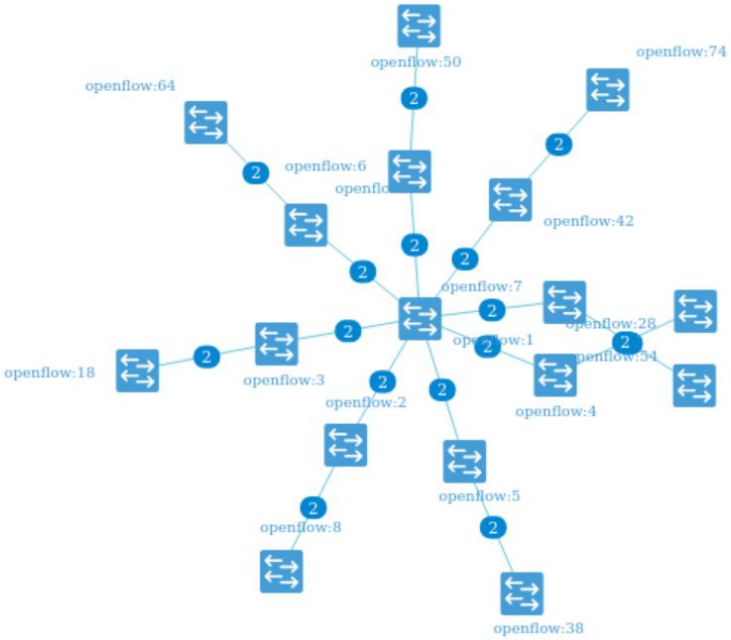


Fuente: Elaboración Propia

OFM (*OpenFlow Manager*), es la aplicación diseñada para operar sobre ODL, con el propósito de configurar las rutas OpenFlow en los dispositivos de red y obtener estadísticas relevantes. Utiliza el protocolo NBI REST para comunicarse con los dispositivos, facilitando la gestión integral de la red desde un único punto de control. En la **Figura 26** se muestran los dispositivos interconectados garantizando que la aplicación OFM, ha identificado y presentado adecuadamente la topología desarrollada en el script de Python.

Figura 26.

API OFM

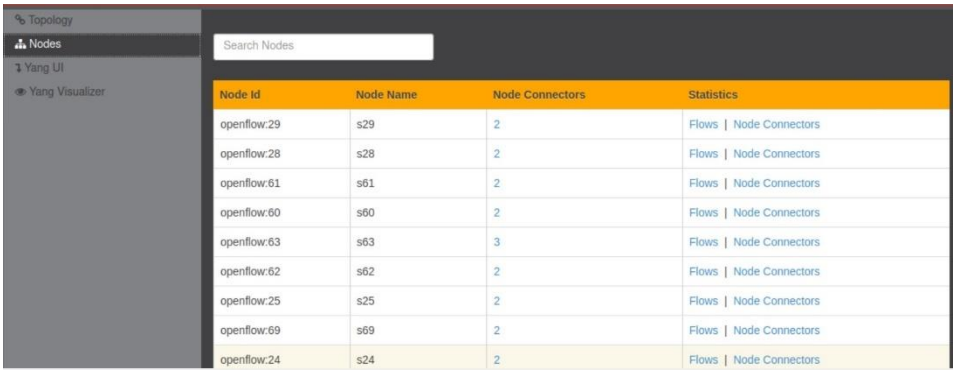


Fuente: Elaboración propia Simulación de Red de Datos/UNL

El controlador OpenDaylight (ODL), mediante los módulos instalados, facilita la visualización detallada de la ruta de los paquetes, incluyendo información sobre la cantidad, errores y direcciones. Para acceder a esta información, es necesario navegar al menú específico del controlador como se muestra en la Figura 27.

Figura 27.

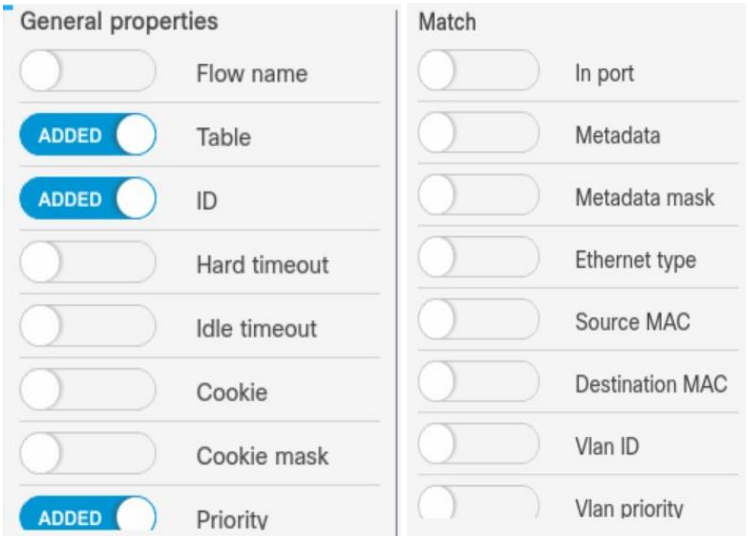
Propiedades generales Controlador



Fuente: Elaboración propia

Las tablas de flujo empleadas en el controlador SDN OpenDaylight (ODL) pueden ser configuradas con políticas que desempeñan diversas funciones en la red, proporcionando beneficios notables en términos de administración, confiabilidad y otras ventajas. Este enfoque permite una gestión más eficiente y personalizada del tráfico en la red, presentados en la Figura 28, adaptando su comportamiento según las necesidades y objetivos específicos del entorno.

Figura 28.
Tabla de funciones



Fuente: Elaboración Propia

6.6 Pruebas de desempeño del diseño de red SDN

Para desarrollar la propuesta de diseño de red basada en la tecnología SDN, se estableció un controlador OpenDaylight como punto central. Este controlador supervisa 8 switches de distribución ubicados en diferentes facultades de la Universidad Nacional de Loja. Cada facultad cuenta con 2 switches de acceso y 4 hosts. Para evaluar la conectividad, se realizaron pruebas mediante "ping" de manera simultánea desde cada host hacia los servidores locales. Estas pruebas generaron tráfico simulado, representativo de una hora pico, mediante el software de simulación, lo que nos proporcionará datos relevantes sobre las métricas de evaluación mencionadas anteriormente, permitiéndonos realizar una comparativa detallada que se presentará más adelante.

El proceso inicia con el envío del paquete por un dispositivo el cual es un mensaje ARP, que es encapsulado por el Switch de capa de acceso y se envía a OpenDayLight mediante “packet_in”. Luego el controlador recepta el paquete y a través de la funcionalidad del módulo de servicio L2-Switch se encarga de interpretar la solicitud.

Posteriormente, el controlador responde con el envío de un mensaje tipo 'packet_out' para enviar mensajes en modo broadcast es decir inundando todos los puertos del conmutador, a excepción del puerto por el cual se recibe el paquete.

Además, el host origen envía los paquetes ICMP con las direcciones IP asignadas del host destino para que el Switch con el protocolo OpenFlow los encapsule y se envía al controlador OpenDayLight a través del mensaje 'packet_in'. Por consiguiente, el controlador recibe el paquete y realiza una búsqueda en su base de datos, la cual tiene como nombre 'Address Tracker' asociado a una de las funcionalidades del módulo L2-Switch, con la finalidad de obtener la ubicación del nodo destino en la topología de red SDN.

El proceso anterior es beneficioso para nuestra red SDN, ya que las tablas de flujo se encontrarán agregadas en los conmutadores OpenFlow y permitirá que los paquetes ICMP se envíen a los diferentes nodos, obteniendo un tiempo de respuesta desde un host origen hacia el host destino menor a los 5 milisegundos. Así mismo, la actualización de las tablas de flujo se realizará de acuerdo con los eventos que se presenten como la inactividad de un puerto de un conmutador, caída de un enlace o nodo e incluso la agregación de nuevos dispositivos en la red SDN

- **Latencia SDN**

Después de finalizar con las pruebas de conectividad desde los hosts hacia los servidores locales, la información obtenida son los tiempos de respuesta en formato de milisegundos (ms). En la Tabla 14 se observa los resultados de los tiempos de respuesta desde los hosts hacia los servidores, con las diferentes longitudes de tramas establecidas como se mencionó anteriormente.

En esta prueba se busca medir los tiempos de respuesta entre los hosts de distintas facultades de la red de manera simultánea, y la pérdida de paquetes.

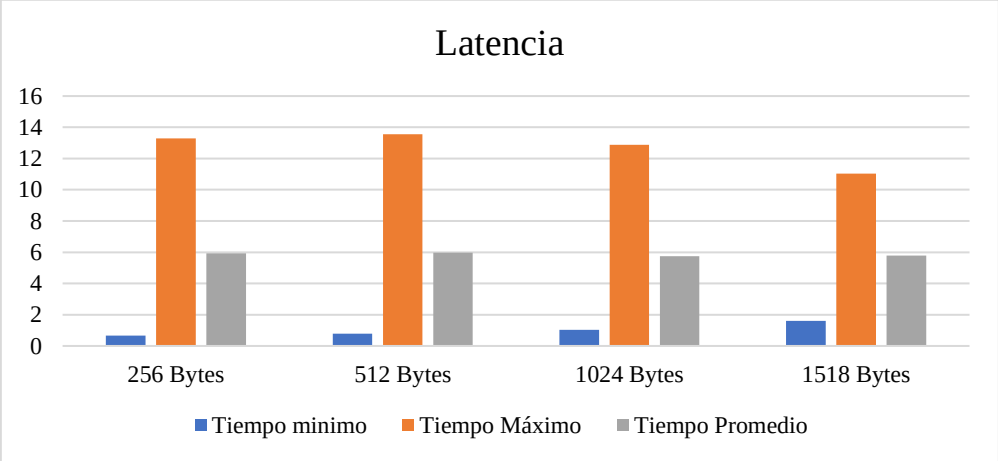
Tabla 14.
Latencia SDN

Longitudes de Paquetes	Tiempo Mínimo	Tiempo Máximo	Tiempo Promedio
256 Bytes	0.67 ms	13.28 ms	5.94 ms
512 Bytes	0.78 ms	13.55 ms	5.97 ms
1024 Bytes	1.04 ms	12.88 ms	5.75 ms
1518 Bytes	1.60 ms	11.04	5.78 ms

Fuente: Elaboración Propia

Una vez finalizadas las pruebas de conectividad ICMP desde los hosts hacia los servidores, se han obtenido resultados que representan la latencia en milisegundos. Estos resultados son evidencia del impacto de la propuesta de SDN en la reducción del tiempo de latencia entre los dispositivos finales y la granja de servidores. Para obtener una visión completa, se consideraron distintas longitudes de tramas, registrando los tiempos mínimos, máximos y promedio de respuesta. Estos datos se presentan de forma visual en la Figura 29. La gráfica de barras muestra claramente cómo varían los tiempos de respuesta en función de la longitud de los paquetes, la barra de color azul representa el tiempo mínimo de latencia, la barra anaranjada representa el tiempo máximo, y la barra de color gris, la latencia promedio, brindando una representación visual intuitiva de los resultados obtenidos del funcionamiento de la red basada en SDN simulada con GNS3.

Figura 29.
Latencia Propuesta SDN

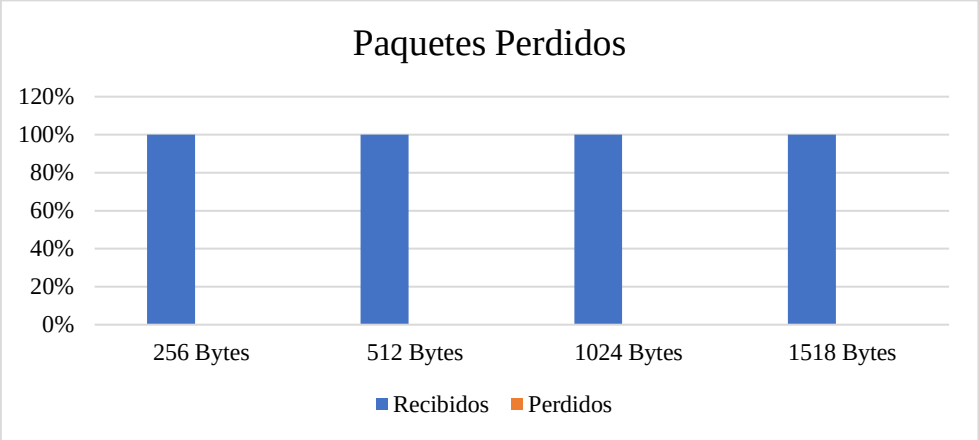


Fuente: Elaboración propia

- **Pérdidas de Paquetes/SDN**

En el indicador del porcentaje de pérdidas de paquetes de los resultados obtenidos, se visualizan en la Figura 30, en la que nos brindan un valor numérico de 0% de pérdidas de paquetes, lo que significa que la conectividad desde los hosts hacia los servidores es exitosa.

Figura 30.
Paquetes perdidos /SDN



Fuente: Elaboración propia

- **Jitter /SDN**

Los resultados obtenidos muestran los tiempos de jitter registrados en la Tabla 15 para las distintas longitudes de paquetes. Se observa que, a medida que aumenta la longitud de los paquetes, el jitter tiende a disminuir. Por ejemplo, para paquetes de 1518 bytes, el jitter es de 6.67 milisegundos, mientras que, para paquetes más pequeños, como los de 256 bytes, el jitter es de 8.91 milisegundos. Esta variación se debe a la forma en que la red maneja los diferentes tamaños de paquetes y cómo se priorizan en el enrutamiento. La variación en los valores de jitter en una red SDN puede atribuirse a la forma en que el controlador gestiona el enrutamiento y procesamiento de los paquetes de datos, especialmente en función de su longitud. El controlador podría optimizar las rutas para paquetes más largos (1024 y 1518 bytes), lo que resulta en una menor variabilidad en los tiempos de entrega y, por ende, en un menor jitter. Sin embargo, para paquetes más cortos, el procesamiento en los dispositivos de red puede ser menos eficiente, lo que aumenta la latencia y, por consiguiente, el jitter.

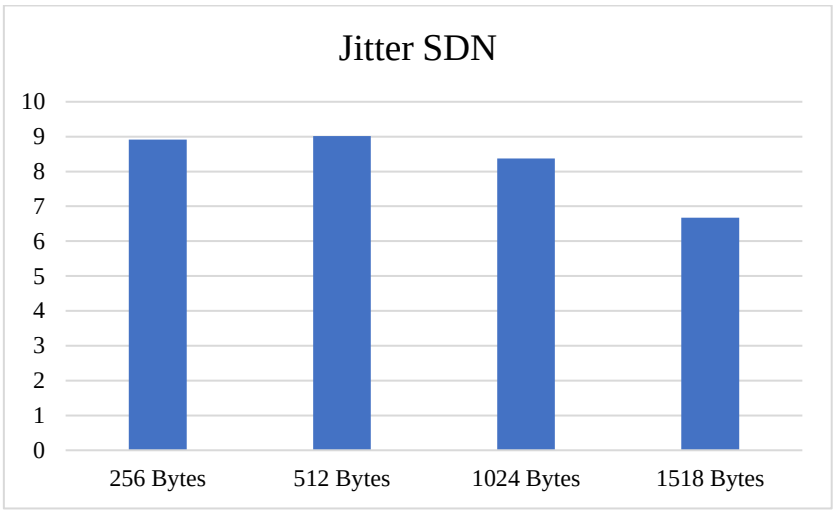
Tabla 15.
Jitter SDN

Longitudes de Paquetes	**JITTER**
256 Bytes	8.91 ms
512 Bytes	9.02 ms
1024 Bytes	8.37 ms
1518 Bytes	6.67 ms

Fuente: Elaboración propia

Para una visualización más clara de estos resultados, se presenta en la Figura 31, una gráfica de barras en la que se muestran los tiempos de jitter para cada longitud de paquete. Esta representación visual permitirá una mejor comprensión de cómo varía el jitter en función del tamaño de los paquetes y cómo impacta en el rendimiento de la red SDN

Figura 31.
Jitter SDN



Fuente: Elaboración propia

6.7 Comparativa De Redes Tradicional /SDN

Tras haber llevado a cabo las pruebas requeridas para obtener los datos relativos a la latencia, el jitter y la pérdida de paquetes de manera independiente, y en el mismo entorno de simulación, procedemos a comparar los resultados entre la red convencional y la red SDN propuesta. Los resultados obtenidos se analizan en conjunto para determinar si existe una mejora significativa o un resultado positivo en la red SDN en comparación con la tradicional.

- **Latencia.**

A continuación, se presenta el análisis de los datos obtenidos, los cuales están clasificados por cada longitud de paquete, se debe tomar en cuenta que se realizó las conexiones a 3 servidores diferentes para obtener datos sobre el funcionamiento de la red lo más acertado posible.

En la Tabla 16 se muestran los valores correspondientes a la media de la latencia generada al realizar conexiones ICMP a los servidores locales, con las distintas longitudes de paquetes, esta idea aplica para ambos paradigmas.

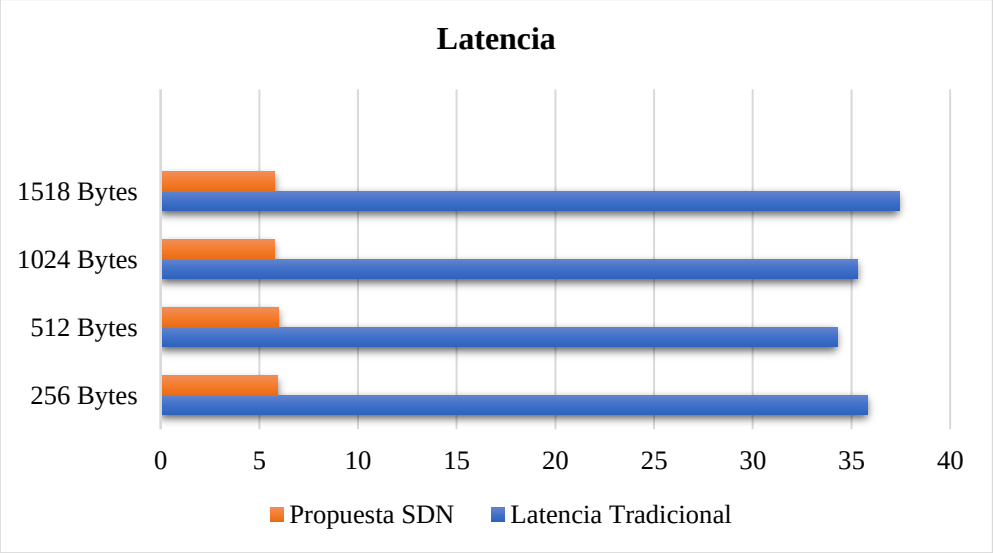
Tabla 16.
Latencia promedio Tradicional/SDN

Longitud de Paquetes	Latencia promedio Red Tradicional	Latencia Promedio SDN
Promedio 256 B	35.81 ms	5.94 ms
Promedio 512 B	34.28 ms	5.97 ms
Promedio 1024 B	35.33 ms	5.75 ms
Promedio 1518 B	37.42 ms	5.78 ms

Fuente: Elaboración propia

En el caso de SDN, el uso del controlador ODL de la red en complemento con la aplicación OFM, para la gestión de las tablas de flujo que se ejecutan en los switches favorece el control y configuración centralizado de la red, además de proveer altos niveles de programabilidad en la misma, esto se evidencia en la Figura 32, en la que se puede observar como el paradigma de red SDN presenta ventajas en cuanto al promedio calculado entre los mejores valores (menores) correspondientes a la latencia.

Figura 32.
Latencia promedio Tradicional/SDN



Fuente: Elaboración propia

- **Jitter**

La Tabla 17 presenta una comparación del jitter promedio entre una red tradicional y una red definida por software (SDN) para diferentes longitudes de paquetes. Se observa que, para paquetes de 1518 bytes, el jitter promedio en la red SDN es considerablemente menor en comparación con la red tradicional, con un valor de 6.67 milisegundos en SDN frente a 8.75 milisegundos en la red tradicional. Asimismo, para paquetes de 1024 bytes,

se registra una mejora en el jitter en la red SDN, con un valor de 8.37 milisegundos en comparación con 8.55 milisegundos en la red tradicional, lo que representa una diferencia de 0.18 milisegundos a favor de la SDN.

Sin embargo, se observa que, para paquetes de 256 y 512 bytes, la red tradicional muestra un jitter promedio menor en comparación con la red SDN. Esto indica que, en el escenario de simulación que trabajamos y referente a los tamaños de paquetes específicos, la red tradicional puede ofrecer un mejor rendimiento en términos de jitter.

Tabla 17.
Jitter Promedio Tradicional/SDN

Longitud de Paquetes	Jitter Promedio Tradicional	Jitter Promedio SDN
Promedio 256 B	8.26 ms	8.91 ms
Promedio 512 B	7.96 ms	9.02 ms
Promedio 1024 B	8.55 ms	8.37 ms
Promedio 1518 B	8.75 ms	6.67 ms

Fuente: Elaboración Propia

La Figura 33 muestra una comparación del jitter promedio entre una red tradicional y una red SDN para diferentes longitudes de paquetes. En el eje vertical se encuentran las longitudes de paquetes, mientras que en el eje horizontal se representa el jitter promedio en milisegundos.

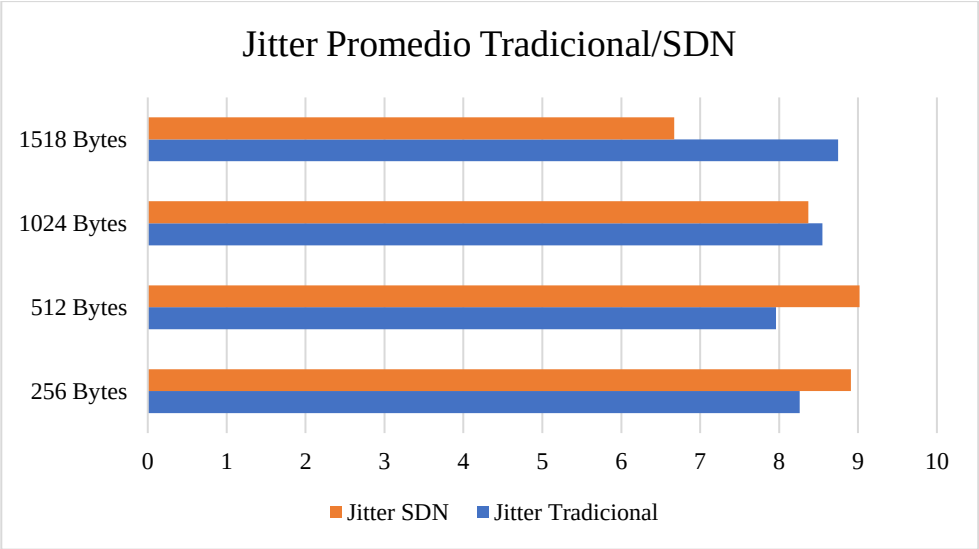
En la gráfica, se pueden observar cuatro conjuntos de barras agrupadas, cada una correspondiente a una longitud de paquete específica. Dentro de cada conjunto, se representan dos barras: una barra de color azul para el jitter promedio en la red tradicional y otra barra color anaranjado para el jitter promedio en la red SDN.

Para las longitudes de paquetes de 1518 y 1024 bytes, se puede notar que el jitter promedio en la red SDN es significativamente menor en comparación con la red tradicional, lo que indica una mejora en la estabilidad y consistencia de la entrega de datos en la red SDN cuando la red está más cargada.

Sin embargo, para las longitudes de paquetes de 256 y 512 bytes, se observa que el jitter promedio en la red tradicional es menor que en la red SDN, debido a la variabilidad que presenta el controlador mencionado anteriormente. Esto sugiere que, en el envío de paquetes con longitudes más cortas, la red tradicional puede ofrecer un rendimiento más consistente en términos de jitter.

Figura 33.

Jitter Promedio Tradicional/SDN



Fuente: Elaboración Propia

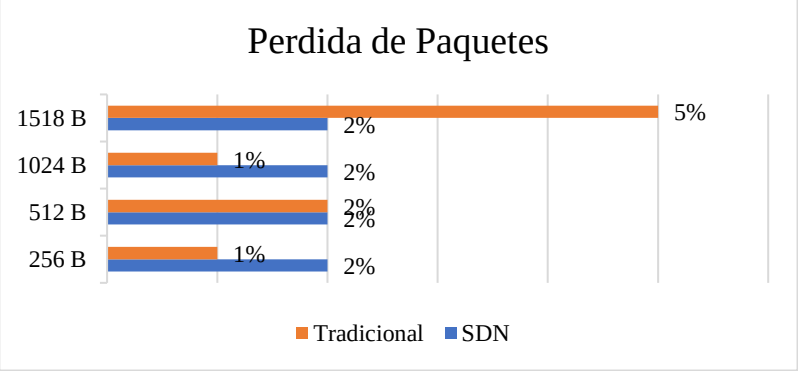
- **Pérdida de Paquetes**

En la Figura 34 se muestran los resultados obtenidos de las pruebas llevadas a cabo en el entorno de estudio. En el gráfico, el color azul representa la red SDN, donde se observa una tasa de pérdida de paquetes del 2%. Esta pérdida se atribuye a la eficiencia de las tablas de flujo en la SDN. En contraste, en la red tradicional, representada en color anaranjado, la tasa de pérdida de paquetes varía entre el 1%, 2% y el 5%, lo que se considera mínima debido a su capacidad de reenvío preciso.

Por lo tanto, se confirma la reducción del porcentaje de pérdida de paquetes al realizar pruebas de conectividad desde los dispositivos finales hacia la granja de servidores, lo que nuevamente contrasta los resultados frente a la red tradicional.

Figura 34.

Pérdida de Paquetes Tradicional /SDN



Fuente: Elaboración Propia

6.8 Presupuesto básico estimado para implantación de una red SDN

Una red híbrida presenta la ventaja de permitir una transición gradual desde la arquitectura de red actual hacia una completamente definida por software.

El siguiente es un presupuesto estimado para dispositivos SDN utilizando productos del proveedor HP. Se seleccionaron equipos de HP debido a sus características técnicas y su capacidad para integrar aplicaciones de terceros. Según (Tienda HPE EE. UU.), la tienda de aplicaciones HP SDN proporciona una plataforma para que los proveedores de software independientes puedan ofrecer soluciones innovadoras, permitiendo a los administradores de TI abordar sus desafíos de red de manera única a través de estas aplicaciones.

El estándar OpenFlow v1.3 incluye especificaciones para la interoperabilidad entre el tráfico OpenFlow y el tráfico no OpenFlow, facilitando así una migración gradual hacia SDN. OpenFlow v1.3 define dos tipos de switches compatibles: OpenFlow puro e híbrido. Los switches híbridos OpenFlow son capaces de admitir tanto OpenFlow como la conmutación tradicional de una red convencional. El proveedor HP ofrece hardware de red que tiene la capacidad de operar en modo híbrido.

- **Especificaciones Técnicas de Equipos**

➢ **CONTROLADOR**

El HP Virtual Application Networks (VAN) SDN Controller ofrece un punto de control centralizado en una red compatible con SDN, simplificando así la gestión, el aprovisionamiento y la orquestación. Esta capacidad facilita la entrega de servicios de red

basados en aplicaciones de próxima generación y proporciona interfaces de programación de aplicaciones (API) abiertas, lo que permite a los desarrolladores crear soluciones innovadoras para adaptar dinámicamente los requisitos empresariales a la infraestructura de red. Estas soluciones pueden desarrollarse mediante programas Java personalizados o mediante interfaces de control RESTful de propósito general.

El controlador HP VAN SDN está diseñado para operar en una variedad de entornos, incluidos campus, centros de datos y proveedores de servicios, y cuenta con las siguientes características principales:

- Procesamiento de flujo proactivo
- Procesamiento de flujo reactivo
- Interfaz gráfica de usuario (GUI)
- API hacia el norte
- Arquitectura escalable
- Alta disponibilidad
- Seguridad del controlado
- Módulo de servicio de enlace
- Módulo de servicio de topología
- Módulo de servicio del administrador de nodo
- Interfaz de control de OpenFlow

- **Switch HP ARUBA 1960 24G/ Capa de Core**

Este dispositivo de red, de tipo apilable y configuración fija, destaca por su excelente rendimiento y su eficiente gestión. Con 24 puertos Gigabit que ofrecen una conectividad incomparable, este switch exhibe su avanzada ingeniería al integrar un núcleo ARMv7 y un potente Cortex-A9 de 800 MHz, otorgándole una distinguida clasificación de Clase 4 y Clase 6. Además, su capacidad PoE simplifica la alimentación directa de dispositivos a través de cables de red, mientras que los puertos SFP+ y una capacidad de conmutación de 128 Gbps garantizan una transmisión de datos fluida a una impresionante velocidad de 95 Mpps. Con un consumo energético eficiente de 370 W, este switch se posiciona como una elección esencial para empresas pequeñas en crecimiento, gracias a su fácil implementación y su asequibilidad.

- **SWITCH HP ARUBA 2930 24G/ Capa de Distribución**

La Serie 2930 ofrece seguridad, escalabilidad y facilidad de uso para redes empresariales de borde, SMB y sucursales. El switch HP 2930 admite velocidad de transferencia de datos de hasta 100 megabits por segundo, con una RAM de 1024 MB.

Principales características:

- Serie de conmutadores Aruba Capa 3 con apilamiento, enrutamiento estático y RIP, IPv6, ACL y sFlow para una mejor experiencia de campus móvil primero
- Enlaces ascendentes modulares de 10 GbE y fuentes de alimentación actualizables.
- Soporte OpenFlow

- **SWITCH HP SWITCH 5400ZL SERIES/Capa de Accesso**

La serie HP Switch 5400zl consta de los conmutadores inteligentes más avanzados de la línea de productos HP ProCurve. La serie 5400zl incluye chasis de 6 ranuras y 12 ranuras y módulos zl y paquetes asociados. Con interfaces 10/100, Gigabit y 10-Gigabit, PoE + integrado en puertos 10/100 y 10/100 / 1000Base-T, y una selección de factores de forma, los switches 5400zl ofrecen una excelente protección de la inversión, flexibilidad y escalabilidad, así como también como facilidad de implementación, operación y mantenimiento.

Principales características:

- Rendimiento y seguridad de clase empresarial
- Ubicación inteligente remota
- Multiples archivos de configuración
- Soporte OpenFlow
- Arquitectura de alta velocidad/capacidad
- Conmutación de capa 2

- **EQUIPOS TERMINALES**

Los equipos terminales tales como los computadores, impresoras, laptops, teléfonos móviles, entre otros; no necesitan características adicionales para integrarse a la red SDN.

- **Presupuesto Estimado**

La cantidad de equipos se calcularon de acuerdo con la topología planteada para la Simulación, se utilizó la herramienta de Obras, para contar con un estimado lo más cercano posible.

CÓDIGO	DESCRIPCIÓN	UNIDAD	CANTIDAD	PRECIO UNITARIO	PRECIO TOTAL
1740	CONTROLADOR	u	1	399.95	399.95
1741	SWITCH CORE	u	1	1,220.00	1,220.00
1742	SWITCHDISTRIBUCIÓN	u	8	1299.99	10,399.92
1743	SWITCH ACCESO	u	100	1,136.00	113,600.00
				Total USD $	**124.400.39**

PRECIO TOTAL DE LA OFERTA: CIENTO VEINTICUATRO MIL CUATROCIENTOS dólares con TREINTA Y NUEVE centavos

Fuente: (HPE (VAN) SDN Controller Base.) (Techno Prime, n.d.) (SWITCH HPE ARUBA JL259A 2930F-24G.) (HP Switch de Red 5400 Zl.)

7. Discusión

El avance en las tecnologías de redes ha sido constante, y las Redes Definidas por Software (SDN) han surgido como una solución prometedora para optimizar mucho más la gestión y el rendimiento de las redes. En este contexto, este trabajo se centra en el desarrollo de una propuesta de diseño basada en SDN para la red de datos de la Universidad Nacional Loja, Campus La Argelia, Ciudad de Loja. Para evaluar la eficacia de esta propuesta, se realizaron simulaciones en GNS3 para implementar la red tradicional de la institución y compararla con la red propuesta SDN realizada con Mininet, implementada en Mininet, con métricas de latencia, pérdida de paquetes y jitter como principales indicadores de desempeño, ya que es la información que actualmente se registra en la UTI.

La red tradicional de la Universidad Nacional Loja consta de una topología jerárquica de tres capas: núcleo, distribución y acceso. En contraste, la propuesta SDN busca mejorar la gestión y la eficiencia mediante la separación del plano de control y el plano de datos, centralizando la gestión de la red en un controlador SDN.

Para garantizar resultados adecuados, se emplearon equipos adecuados y bien caracterizados generando flujos de tráfico simulados para evaluar el rendimiento de ambas redes en términos de latencia, pérdida de paquetes y jitter.

Los resultados obtenidos revelaron una mejora en el rendimiento de la red con la implementación de SDN. En términos de latencia, con los datos obtenidos se observó una reducción del 19.10 %, porcentaje significativo en comparación con la red tradicional. Esto se atribuye a la capacidad de SDN para tomar decisiones de enrutamiento de manera más eficiente y dinámica. A sí mismo, en cuanto a la pérdida de paquetes, la red SDN mostró un porcentaje mínimo del 2 %, en comparación con la red tradicional que fue variado del 1,2 y 5%. Esto se debe a la capacidad del controlador SDN para gestionar el tráfico y evitar congestiones mediante la asignación dinámica de recursos.

Por último, el jitter, que mide la variación en el retardo de la transmisión de datos, también se redujo un 6% en la red SDN. Esto se debe a la capacidad de SDN para optimizar el enrutamiento y la asignación de recursos de manera más eficiente, lo que resulta en una transmisión de datos más estable y predecible.

Los resultados de la simulación confirman que la propuesta de diseño basada en SDN para la red de datos de la Universidad Nacional Loja ofrece mejoras prometedoras

en términos de latencia, pérdida de paquetes y jitter en comparación con la red tradicional. Estos hallazgos respaldan la viabilidad y la eficacia de la implementación de SDN en entornos universitarios, siempre y cuando se utilicen equipos adecuados y se realice una buena caracterización de la red. Por lo tanto, se recomienda que futuras investigaciones consideren la implementación práctica de la propuesta SDN en el entorno real de la Universidad Nacional Loja para validar los resultados de la simulación.

8. Conclusiones

- La investigación de los conceptos fundamentales de las redes definidas por software (SDN) permitió una comprensión profunda de los principios que sustentan esta tecnología. Se exploraron aspectos como la separación del plano de control y el plano de datos, la centralización de la gestión y la programabilidad de la red. De la misma forma, se examinó en detalle la arquitectura, protocolos y otros componentes que conforman las SDN, lo que posibilitó identificar características y ventajas situando su potencial impacto en la eficiencia de las redes en comparación con las redes tradicionales.
- Se diseñó una propuesta de optimización basada en SDN sobre el estado actual de la red de datos de la Universidad Nacional de Loja (UNL) que se elaboró considerando los desafíos y necesidades específicas de la institución. La identificación de protocolos, controladores y emuladores ayudó a establecer con qué herramientas de software se puede contar y cuáles son las que se acoplan a los requerimientos de esta investigación. Se propuso la implementación de SDN utilizando el controlador OpenDayLigth y el protocolo Openflow, con el objetivo de mejorar la eficiencia y flexibilidad, manteniendo una correcta interacción entre los dispositivos y la administración de las redes. Esta propuesta se fundamenta en el análisis de la infraestructura de red existente en la UNL.
- La evaluación de la propuesta planteada se realizó mediante simulaciones utilizando los simuladores Mininet y GNS3, software que brindan las prestaciones adecuadas para permitir crear diferentes tipos de topologías de red y la realización de las evaluaciones correspondientes con las métricas establecidas para la comparación del rendimiento de la red tradicional con la implementación SDN propuesta.
- En el presente caso de estudio ha permitido realizar la comparación entre los paradigmas de una SDN centralizada y una red tradicional jerárquica. Los resultados del rendimiento de ambos paradigmas muestran cómo la red definida por software presenta una reducción en la latencia de la transmisión del paquete de datos del 19.10%, presentando mejor rendimiento que la red tradicional, para el caso del escenario analizado, por lo que la propuesta es factible a implementar.

- La red de datos de la UNL cumple con los estándares establecidos en el rendimiento de red; pero la implementación de tecnologías avanzadas como SDN, que ofrecen una latencia significativamente reducida, podría mejorar la experiencia del usuario en aplicaciones interactivas y optimizar el rendimiento de aplicaciones críticas y sensibles a la latencia; de la misma forma la adopción de SDN prepararía la infraestructura de red para futuras demandas tecnológicas y del mercado, asegurando que la red de la UNL se mantenga competitiva y adaptable. Esto no solo incrementa la eficiencia operativa general de la misma, sino que también posiciona a la universidad a la vanguardia de la innovación tecnológica en la gestión de redes.

9. Recomendaciones

- Si se requiere implementación de infraestructura SDN, se debe considerar que los equipos de red heredados estén actualizados con la última versión de software; lo que permitirá desplegar SDN de forma gradual como una red híbrida. En redes híbridas los dispositivos se pueden ir reemplazando parcialmente, dado que, el controlador debe tener comunicación con el plano de control de los dispositivos heredados.
- Fomentar la capacitación del personal de TIC y los investigadores de la UNL en las nuevas tecnologías y conceptos asociados con SDN, proporcionando escenarios prácticos en colaboración con otros expertos y profesionales facilitando el intercambio de conocimientos y experiencias, impulsando la innovación y el progreso en el ámbito de las redes definidas por software.
- Se recomienda seguir explorando y desarrollando casos de uso específicos de SDN que sean relevantes para el entorno académico de la UNL. Podría incluir aplicaciones como la gestión automatizada de recursos de red para laboratorios o la provisión dinámica de servicios para aulas virtuales.
- Se recomienda investigar técnicas de segmentación de red y microsegmentación para mejorar la seguridad y la gestión de recursos en redes SDN, realizando pruebas y monitoreo continuo para maximizar estos beneficios y mantener la red optimizada, asegurando así un rendimiento superior y una gestión efectiva de la calidad de servicio (QoS).
- Es necesario continuar con más proyectos respecto a esta nueva tecnología de las redes definidas por software, con la finalidad de continuar con la actualización técnica y académica de este tema para beneficio de toda la comunidad universitaria.

10. Bibliografía

Aguirre, X. F., & Crespo, M. M. (2021). *"CONVERGENCIA DE REDES DEFINIDAS POR SOFTWARE CON OPENDAYLIGHT Y NSX SOBRE UNA NUBE PRIVADA."*

Alava, C., & Paladines, D. (2020). *"DISEÑO E IMPLEMENTACIÓN DE UN MÓDULO DIDÁCTICO DE RED DEFINIDA POR SOFTWARE (SDN) PARA PRÁCTICAS UNIVERSITARIAS CON PROTOCOLO OPENFLOW MEDIANTE HARDWARE LIBRE."* UNIVERSIDAD POLITÉCNICA SALESIANA SEDE GUAYAQUIL .

Cáceres, J., & Casilimas, C. (2022). *Arquitectura y funcionamiento de redes definidas por software (SDN)*. https://repository.udistrital.edu.co/handle/11349/29727

Chafloque, J. (2018). *Propuesta de diseño de una red de datos de área local bajo la arquitectura de redes definidas por software para la Red Telemática de la Universidad Nacional Mayor de San Marcos*. https://cybertesis.unmsm.edu.pe/handle/20.500.12672/10017

Domínguez Pérez, J. (2021). *Software Defined Networking: The future of Networks* [Universidad Politécnica de Madrid]. https://oa.upm.es/67923/

EstiNet - Simulator | EstiNet. (n.d.). Retrieved December 6, 2023, from http://www.estinet.com/ns/

GNS3. (n.d.). *GNS3*. Retrieved December 10, 2023, from https://docs.gns3.com/docs/

Gómez, I. (2016). *Arquitectura SDN.Investigación de Redes Definidas por Software Página 15 de 106*. https://silo.tips/download/3-arquitectura-sdn-philosophi-n-a-t-u-r-a-l-i-s-p-r-i-n-c-i-p-i-a-technologica-i

Graf, T. (2013). *Underneath OpenStack Quantum: Software Defined Networking with Open vSwitch*. https://blog.zhaw.ch/icclab/files/2013/04/OpenStack-Quantum-SDN-with-Open-vSwitch.pdf

Guanoluisa, E. (2019). *DISEÑO DE LA ARQUITECTURA DE UNA RED SDN MEDIANTE EL PROTOCOLO OPENFLOW CON SIMULACION EN EL SOFTWARE MININET PARA LA INFRAESTRUCTURA DE UNA PYMES* [UDLA]. https://dspace.udla.edu.ec/bitstream/33000/10884/1/UDLA-EC-TIRT-2019-05.pdf

Guevara, J., & Quizhpi, D. (2017). *DISEÑO DE LA RED DE CAMPUS DE LA EMPRESA "EQUIPOS Y SUMINISTROS DE TELECOMUNICACIONES*

EQUYSUM" DE LA CIUDAD DE QUITO [UNIVERSIDAD POLITÉCNICA SALESIANA SEDE QUITO]. https://dspace.ups.edu.ec/bitstream/123456789/14613/1/UPS%20-%20ST003251.pdf

HP 5412 zl Switch – Switch de red 5400 zl. (n.d.). Retrieved May 13, 2024, from https://www.inovamusicnet.com/producto/hp-5412-zl-switch-switch-de-red-gestionado-montaje-en-bastidor-con-licencia-de-hp-5400-zl-switch-premium-%F0%9F%A5%87%E2%9C%94/

HPE Virtual Application Networks (VAN) SDN Controller Base SW E (J9863AAE). (n.d.). Retrieved May 13, 2024, from https://www.connection.com/product/hpe-virtual-application-networks-van-sdn-controller-base-sw-e-ltu/j9863aae/16766908

IBM. (n.d.). *¿Qué es la latencia?* Retrieved May 9, 2024, from https://www.ibm.com/mx-es/topics/latency

Intriago, W. (2017). *Estudio del protocolo Openflow usando el modelo de red definida por Software (Software Define Networks).* http://repositorio.puce.edu.ec:80/handle/22000/14424

Introducción a Mininet · mininet/mininet Wiki · GitHub. (n.d.). Retrieved December 6, 2023, from https://github.com/mininet/mininet/wiki/Introduction-to-Mininet#what

IONOS. (n.d.). *¿Qué es el jitter? Definición y explicación*. Retrieved May 9, 2024, from https://www.ionos.es/digitalguide/servidores/know-how/jitter/

Jácome, D., & Naranjo, J. (2022). *DISEÑO DE UNA RED PARA EL USO DE APLICACIONES DE VIDEO MEDIANTE REDES DEFINIDAS POR SOFTWARE (SDN)*. http://dspace.ups.edu.ec/handle/123456789/23451

Lasso Guamàn, D. J., & Puchaicela, J. R. (2021). *EVALUACIÒN DEL RENDIMIENTO DE UN PROTOTIPO SDN(SOFTWARE DEFINED NETWORKING) BAJO EL PROTOCOLO OPENFLOW UTILIZANDO HERRAMIENTAS OPEN SOURCE EN UN ENTORNO VIRTUALIZADO*.

Lopera, J., Ramírez Gómez, C., Zuluaga, M., & Ortiz Vanegas, J. (2010). *EL MÉTODO ANALÍTICO COMO MÉTODO NATURAL*. *1*.

Maggio, A. (2018). *Presentamos Cisco PPDIOO para diseño de redes - ICTHore.com*. https://www.ictshore.com/network-design/cisco-ppdioo/

Mc Cabe, & James D. (1998). *Practical Computer Network Analysis and Design*. https://archive.org/details/practicalcompute0000mcca

MD-SAL — Documentación MD-SAL 11.0.6-SNAPSHOT. (n.d.). Retrieved January 24, 2024, from https://docs.opendaylight.org/projects/mdsal/en/latest/index.html

MiniEdit 2.1.0.8 | Tech and Trains. (n.d.). Retrieved December 6, 2023, from https://techandtrains.com/2014/02/07/miniedit-2-1-0-8/

MININET - Fundación de redes abiertas. (n.d.). Retrieved December 6, 2023, from https://opennetworking.org/mininet/

Mininet: una red virtual instantánea en su computadora portátil (u otra PC) - Mininet. (n.d.). Retrieved December 6, 2023, from http://mininet.org/

ONOS. (n.d.). *Controlador SDN de sistema operativo de red abierta (ONOS)*. Retrieved December 5, 2023, from https://opennetworking.org/onos/

Open Networking Foundation. (n.d.). Retrieved November 21, 2023, from https://opennetworking.org/

Open vSwitch. (n.d.). Retrieved December 10, 2023, from https://www.openvswitch.org/

Oviedo, B., Zhuma, E., Bowen, G., & Patiño, B. (2021). *DE UNA RED DEFINIDA POR SOFTWARE QUE PERMITA BRINDAR SERVICIO DE VoIP SEGUROS*. https://orcid.org/0000-0002-5366-5917

ovsdb — Abrir la documentación de vSwitch 3.2.90. (n.d.). Retrieved December 5, 2023, from https://docs.openvswitch.org/en/latest/ref/ovsdb.7/

Pacuar, A. P. (2022). *ANÁLISIS E IMPLEMENTACIÓN DE UN PROTOTIPO DE RED SDN EN EL CAMPUS NORTE DE LA UNACH*. UNIVERSIDAD NACIONAL DE CHIMBORAZO .

Padròn Pèrez, J. D. (2020). *Implementaciòn de una plataforma redundante de control SDN-IoT*. Universidad de Las Palmas de Gran Canaria.

Planas, A. L. (2016). *CONFIGURACIÓN DE UN ENTORNO DE EMULACIÓN QUE PERMITA EL DISEÑO, DESARROLLO Y EVALUACIÓN DE SOFTWARE-DEFINED NETWORKS CON CALIDAD DE SERVICIO*.

Politécnica, E. (n.d.). *UNIVERSIDAD DE EXTREMADURA*.

¿Qué es NS-3? ns-3. (n.d.). Retrieved December 6, 2023, from https://www.nsnam.org/about/what-is-ns-3/

RFC 3746 - Marco de separación de elementos de control y reenvío (ForCES). (n.d.). Retrieved December 5, 2023, from https://datatracker.ietf.org/doc/html/rfc3746

RFC 6241 - Protocolo de configuración de red (NETCONF). (n.d.). Retrieved December 4, 2023, from https://datatracker.ietf.org/doc/html/rfc6241

Ruipérez, J. (2021). *Seguridad en Redes definidas por software (SDN)*. www.etsit.upv.es

Salazar, G. D. (2021). *Hybrid Networking SDN y SD-WAN: Interoperabilidad de arquitecturas de redes tradicionales y redes definidas por software en la era de la digitalización* [Universidad Nacional de La Plata]. https://doi.org/10.35537/10915/129910

Sameer, M., & Goswami, B. (2018). Experimenting with ONOS scalability on software defined network. *Journal of Advanced Research in Dynamical and Control Systems*. https://www.jardcs.org/backissues/abstract.php?archiveid=6460

Santisteban, B. (2020). *Arquitecturas de redes de computadoras definidas por software: revisión bibliográfica*.

Serie SDN, cuarta parte: Ryu, un controlador SDN de código abierto con numerosas funciones compatible con NTT Labs: la nueva pila. (n.d.). Retrieved December 6, 2023, from https://thenewstack.io/sdn-series-part-iv-ryu-a-rich-featured-open-source-sdn-controller-supported-by-ntt-labs/

Servicios de autenticación, autorización y contabilidad (AAA): guía del usuario: documentación maestra de AAA. (n.d.). Retrieved January 24, 2024, from https://docs.opendaylight.org/projects/aaa/en/latest/user-guide.html

SWITCH HPE ARUBA JL259A 2930F-24G ADMINISTRABLE L3 DE 24 PUERTOS GIGABIT 10/100/1000 + 4 PUERTOS GIGABIT O SFP RACKEABLE | TECNIT. (n.d.). Retrieved May 13, 2024, from https://tecnit.com.ec/producto/switch-hpe-aruba-jl259a-2930f-24g-administrable-l3-de-24-puertos-gigabit-10-100-1000-4-puertos-gigabit-o-sfp-rackeable/

Techno Prime. (n.d.). *Switch Aruba Instant On 1960, 24 G, Core ARMv7, Cortex-A9, 800MHz, Clase 4, Clase 6, PoE, SFP+, 370 W, Velocidad 95 Mpps, Conmutación 128 Gbps, JL807A*. Retrieved May 13, 2024, from https://technoprimec.com/product/switch-aruba-instant-on-1960-24g-core-armv7-cortex-a9-800mhz-clase4-clase6-poe-sfp-370w-velocidad-95-mpps-conmutacion-128gbps-jl807a/

Tienda HPE EE. UU. (n.d.). Retrieved May 12, 2024, from https://buy.hpe.com/us/en

11. Anexos

Anexo 1.- Instalación de Software

1.1 Instalación de Mininet

Para comenzar la instalación del software, se aconseja obtener la aplicación desde la página oficial http://mininet.org (añadir enlace). Se sugiere llevar a cabo una instalación nativa directamente desde la fuente, dado que esta opción se adapta de manera más precisa a nuestro sistema operativo. Al utilizar Ubuntu, el procedimiento implica la apertura de una terminal y la ejecución del siguiente comando:

git clone git://github.com/mininet/mininet

Después de haber completado la descarga, procedemos a ingresar al directorio utilizando el siguiente comando:

cd mininet

Dentro de la carpeta a la que hemos accedido, ahora procedemos a seleccionar la instalación de la última versión de Mininet mediante los siguientes comandos:

git tag #list avaiable versions

git checkout -b 2.2.2

Estos comandos permiten visualizar las versiones disponibles y seleccionar específicamente la versión 2.2.2 para la instalación. Con ello, nos aseguramos de utilizar la versión deseada de Mininet en nuestro entorno.

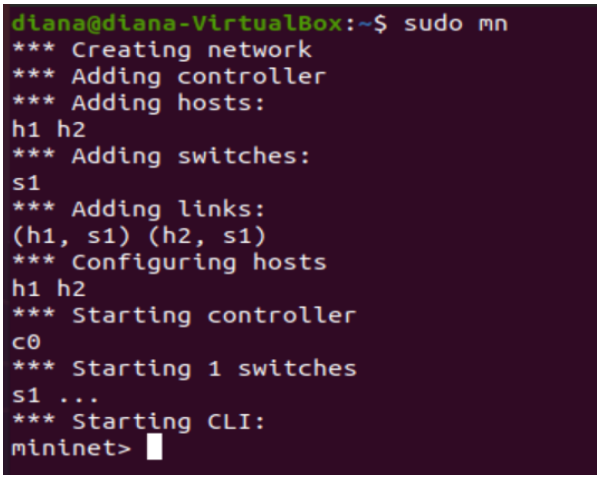
El siguiente paso reviste gran importancia, ya que nos permite instalar todos los elementos ofrecidos por Mininet. Para ello, nos dirigimos a la carpeta "útil" en el repositorio de Mininet mediante los siguientes comandos:

cd útil

mininet/útil/install.sh -a

Con esta secuencia de comandos, se ejecuta el script de instalación, asegurando la instalación completa de Mininet junto con todas sus dependencias y componentes adicionales.

Para crear una topología básica, basta con ingresar el comando “sudo mn”, se crea automáticamente lo siguiente:



Para la creación de una topología personalizada, adaptada a nuestros requerimientos, es esencial desarrollar o modificar un **script en Python**. Con este propósito, se accede al directorio de Mininet, específicamente al subdirectorio de personalización (custom). En este directorio, hemos creado un script para nuestra simulación, el cual presenta características esenciales y comandos que se detallan a continuación:

- **Topo:** indica que se va a crear una topología en mininet
- **addHost:** agrega host a la topología
- **addLink:** agrega los enlaces entre los nodos a la topología
- **addNode:** sirve para el mapeo de puertos de conexión
- **addSwitch:** sirve para agregar switchs a la topología

En la figura # se presenta un ejemplo que viene por defecto en el directorio de la descarga de mininet, en el que se conectan dos conmutadores directamente, con un host en cada conmutador.

```
from mininet.topo import Topo
from mininet.topo import SingleSwitchTopo
from mininet.net import Mininet
from mininet.cli import CLI
from mininet.node import *

class MyTopo( Topo ):
    "Simple topology example."

    def __init__( self ):
        "Create custom topo."

        # Initialize topology
        Topo.__init__( self )

        # Add hosts and switches
        leftHost = self.addHost( 'h1' )
        rightHost = self.addHost( 'h2' )
        newHost1 = self.addHost('h3')
        newHost2 = self.addHost('h4')
        leftSwitch = self.addSwitch( 's3' )
        rightSwitch = self.addSwitch( 's4' )

        # Add links
        self.addLink( leftHost, leftSwitch )
        self.addLink( leftSwitch, rightSwitch )
        self.addLink( leftSwitch, rightHost )
        self.addLink( rightSwitch, newHost1)
        self.addLink( rightSwitch, newHost2)

topos = { 'mytopo': ( lambda: MyTopo() ) }
```

Estructura básica para ejecutar mininet via su API en Python.

Para ejecutar el script, se lo realiza con comandos como el siguiente:

$ sudo mn --custom /mininet/custom/topo-2sw-2host.py

--topo mytopo --test pingall

1.2 Instalación OpenDayLigth (ODL)

Para poder trabajar con el controlador ODL, se debe instalar Java, ya que este controlador es un programa Java. Para realizar lo antes mencionado se procede a usar los siguientes comandos en el terminal de Ubuntu:

Sudo apt-get update

Sudo apt-get install default-jre-headless

Se descarga un archivo denominado ".bashrc", que debemos editar utilizando el comando:

nano bashrc

Se accede a un script Python, y al final del mismo, se añade la siguiente línea de código:

export JAVA_HOME=/usr/lib/jvm/default-java

Este paso permite configurar la variable de entorno JAVA_HOME para apuntar al directorio de instalación de Java necesario para el correcto funcionamiento del controlador ODL.

Una vez instalado Java, procedemos a instalar ODL ingresando al enlace https://nexus.opendaylight.org/content/repositories/public/org/opendaylight/integration/distribution-karaf/0.3.4-Lithium-SR4/. Al acceder a este enlace, el controlador se descarga automáticamente. Posteriormente, extraemos el archivo utilizando el comando:

tar -xvf distribution-karaf-0.3.4-Lithium-SR4.zip

Este comando genera una carpeta llamada "distribution-karaf-0.3.4-Lithium-SR4" que contiene tanto el software como los diversos plugins de ODL. Cabe destacar que Karaf actúa como un contenedor tecnológico que permite a los desarrolladores consolidar todo el software en un único directorio.

Con el siguiente comando arranca ODL:

cd distribution-karaf-0.3.4-Lithium-SR4.

./bin/karaf

Y se presenta inicializado de la siguiente manera:

Para el funcionamiento correcto del controlador se requiere instalar de algunos complementos, los cuales se introducen en la ventana de comando de ODL:

Feature:install odl-restconf /permite el acceso a RESTCONF API

Feature:install odl-l2switch-switch /funcionalidades del switch

Feature:install odl-mdsal-apidocs /acceso a Yang API

Feature:install odl-dlux-all /interfaz grafica

Los comandos previos son esenciales y necesarios para el desarrollo del presente proyecto. No obstante, si se requiere más funcionalidad, se pueden listar las características disponibles mediante el siguiente comando:

feature:list

Para verificar las características instaladas, se utiliza el comando:

feature:list --installed

Una vez contando con las características instaladas, se busca la dirección en la que se encuentra nuestro controlador mediante:

ifconfig

La dirección resultante es la que se utilizará para acceder al controlador a través del navegador. En nuestro caso es:

http://10.0.2.15/index.html#/login

Antes de ingresar a nuestro navegador, no olvidemos ejecutar el ODL, desde el CLI de Linux, una vez ingresada a la carpteda de ODL, con el comando:

./bin/karaf

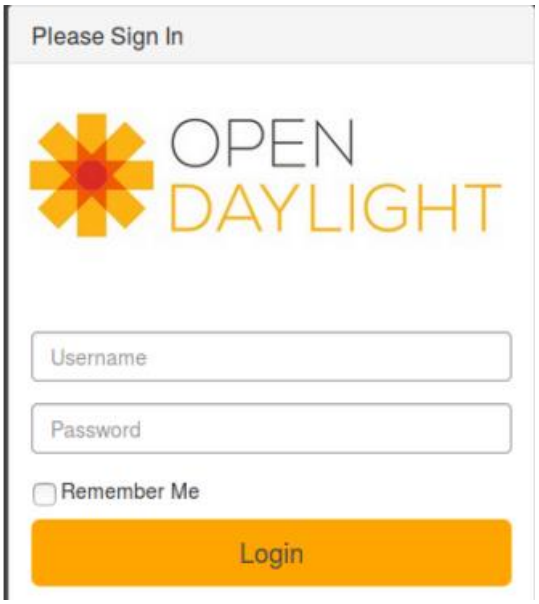
```
diana@diana-VirtualBox:~$ sudo su
[sudo] contraseña para diana:
root@diana-VirtualBox:/home/diana# cd distribution-karaf-0.3.
4-Lithium-SR4
root@diana-VirtualBox:/home/diana/distribution-karaf-0.3.4-Li
thium-SR4# ./bin/karaf
OpenJDK 64-Bit Server VM warning: ignoring option MaxPermSize
=512m; support was removed in 8.0
```

El proceso nos redirige a la interfaz web de ODL, donde se nos solicita ingresar un nombre de usuario y contraseña. Por defecto, las credenciales son las siguientes:

Usuario: admin

Contraseña: admin

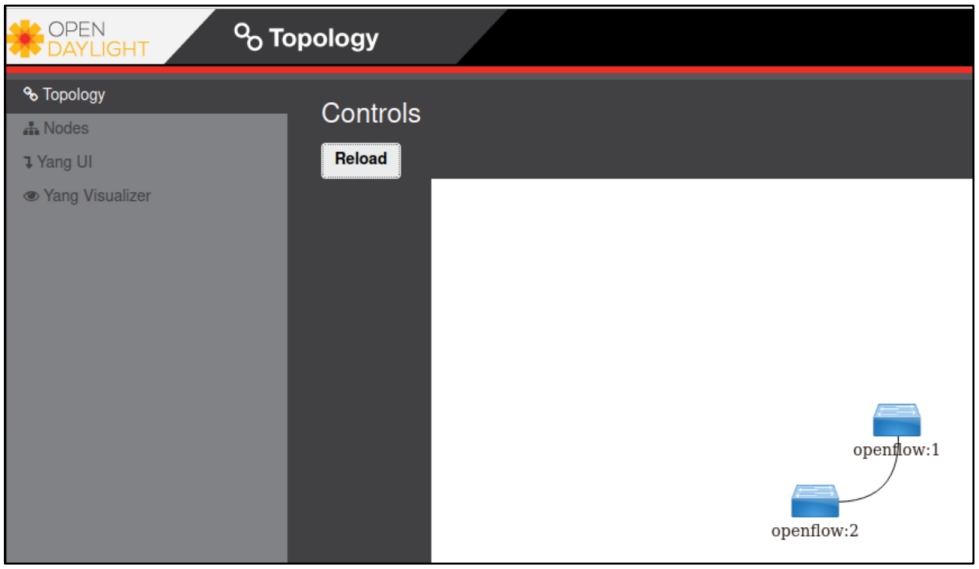
Estas son las credenciales predeterminadas que se utilizan para acceder a la interfaz web de OpenDaylight. Es recomendable cambiar estas credenciales por motivos de seguridad una vez que se haya iniciado sesión por primera vez.



Una vez configurado el controlador, se establece la interconexión con Mininet para observar el funcionamiento básico del controlador. Esto se logra ejecutando el siguiente comando:

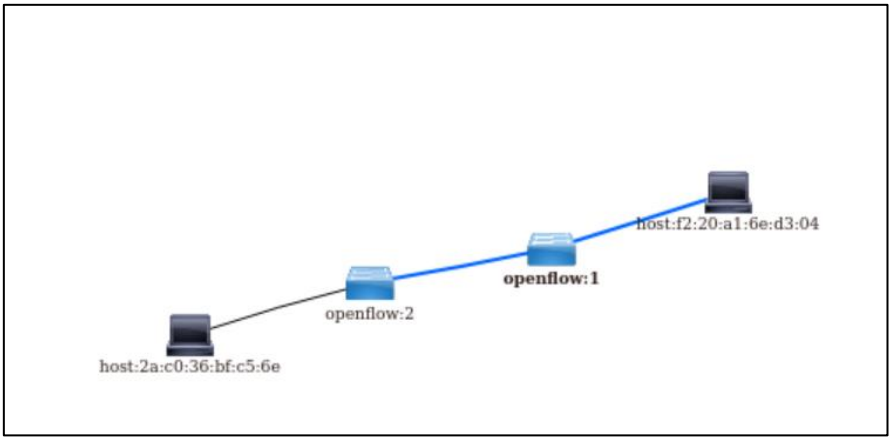
sudo mn --controller=remote, ip=10.0.2.15 --topo single,2 -- switch=ovsk, protocols=OpenFlow10

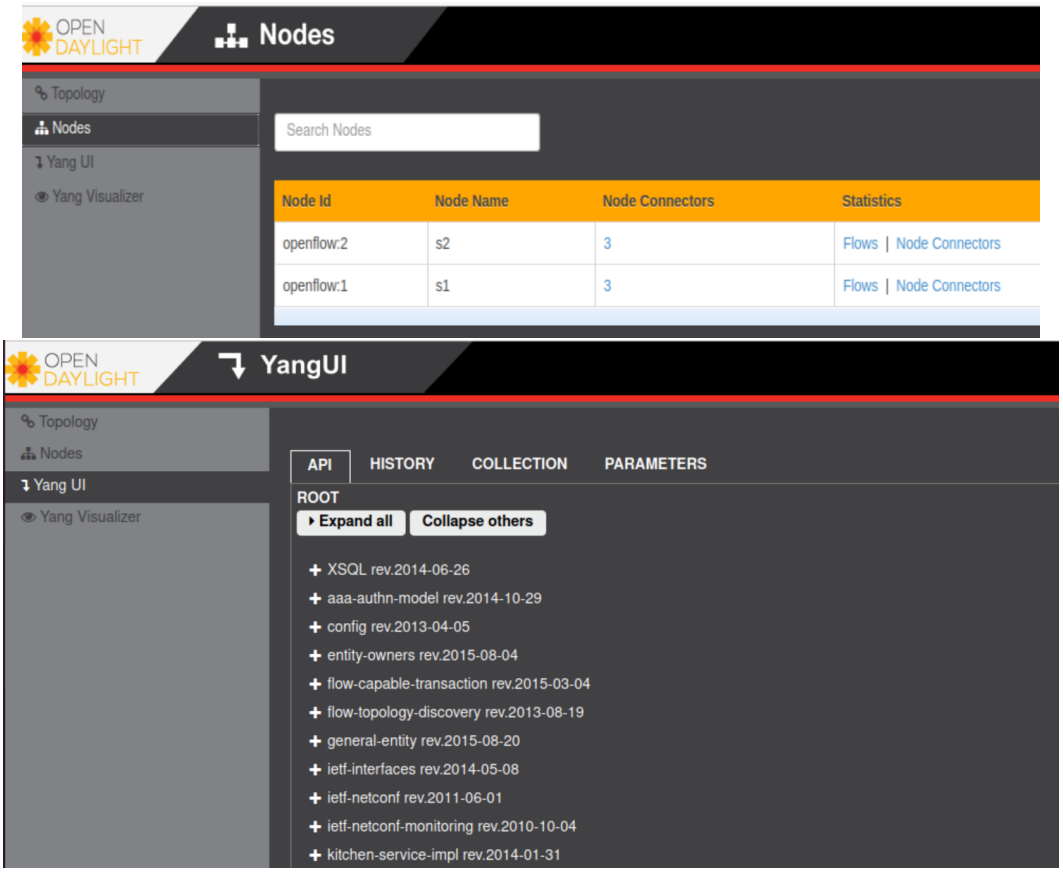
La **interfaz web**, se presenta los siguientes complementos:



Enviando tráfico en la red se visualizan los hots conectados:

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10 h11 h12
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10 h11 h12
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10 h11 h12
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10 h11 h12
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10 h11 h12
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10 h11 h12
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10 h11 h12
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h11 h12
h11 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h12
h12 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11
*** Results: 0% dropped (132/132 received)
mininet>
```





1.3 Instalación de OFM

La API (OpenFlow Manager - OFM) fue instalada en el mismo sistema operativo donde configuramos el controlador para asegurar una comunicación sin errores entre ambos. Dentro de Ubuntu, es necesario instalar algunos componentes previos para garantizar el funcionamiento adecuado de OFM. Para ello, ingresamos los siguientes comandos mediante la interfaz de línea de comandos (CLI)

sudo apt update

sudo apt-get install -y npm

sudo apt-get install -y nodejs-legacy

Se descargo la API OFM desde GitHub a través del CLI de Ubuntu, con el siguiente comando:

git clone http://github.com/CiscoDevNet/OpenDaylight-OpenFlow-App.git

Con la API descargada, es imperativo acceder al archivo llamado env.module.js. Este paso es esencial para definir cómo la API se conectará al controlador, especificando la dirección IP correspondiente.

Posteriormente, como última etapa, se procede a instalar un servidor web llamado Grunt, desde el cual se inicializa la API OFM. Estos pasos se realizan de la siguiente manera:

sudo npm install -g grunt-cli/ Instalacion del servidor grunt

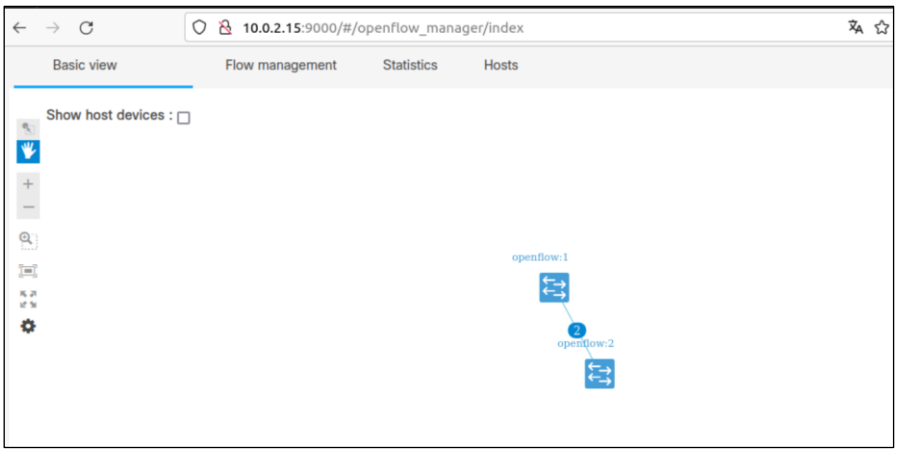
sudo grunt/ ejecución de la API

Estos comandos aseguran la instalación del servidor web Grunt y la ejecución de la API OFM, permitiendo así una correcta interacción entre la API y el controlador, presentando lo siguiente:

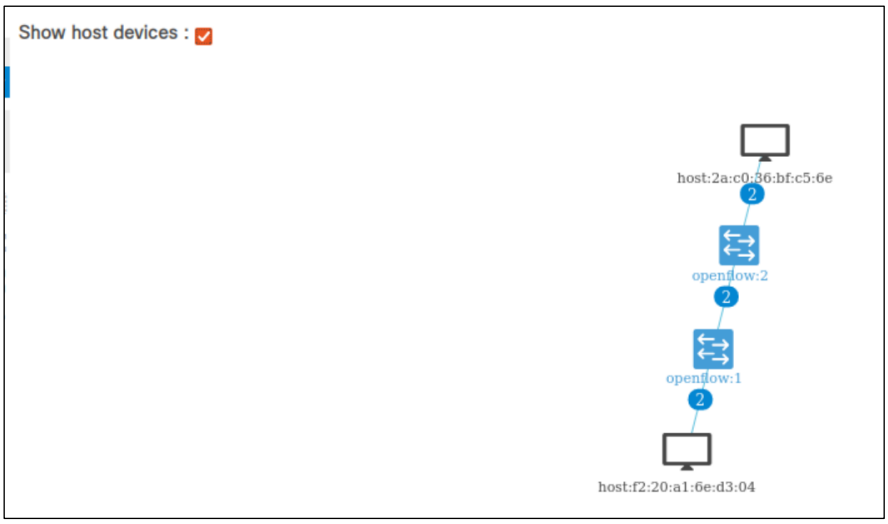
```
root@diana-VirtualBox:/home/diana# cd OpenDaylight-OpenFlow-App/
root@diana-VirtualBox:/home/diana/OpenDaylight-OpenFlow-App# sudo grunt
Running "connect:def" (connect) task
Waiting forever...
Started connect web server on http://localhost:9000
```

Una vez completada la instalación, se puede acceder a la interfaz gráfica de la API OFM a través del navegador. Esto se logra conectándose a la dirección IP de Ubuntu, junto con el controlador, utilizando el puerto 9000 del servidor Grunt.

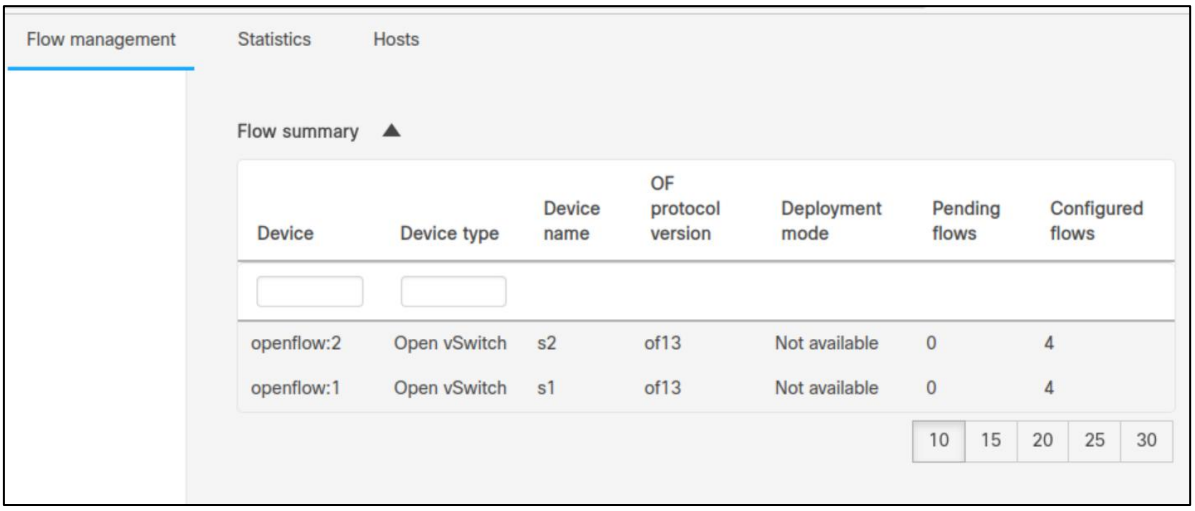
Se visualiza en el servidor web con el siguiente link:

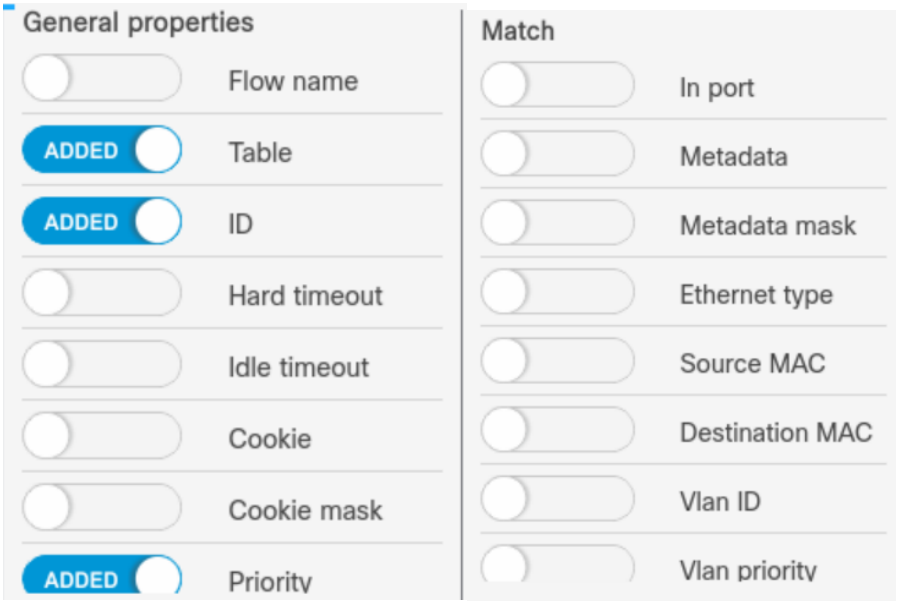


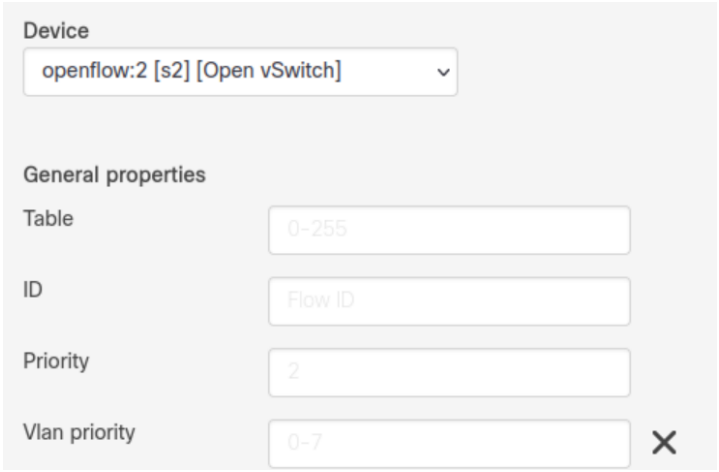
Enviando tráfico se presenta los hots conectados:



La API OFM, nos permite poder crear y editar tablas de flujo, para poder especificar todos los requerimientos o cambios que queramos establecer en la red.







1.4 Instalación de Wireshark

Para instalar Wireshark en Ubuntu a través de la línea de comandos (CLI), puedes utilizar los siguientes comandos:

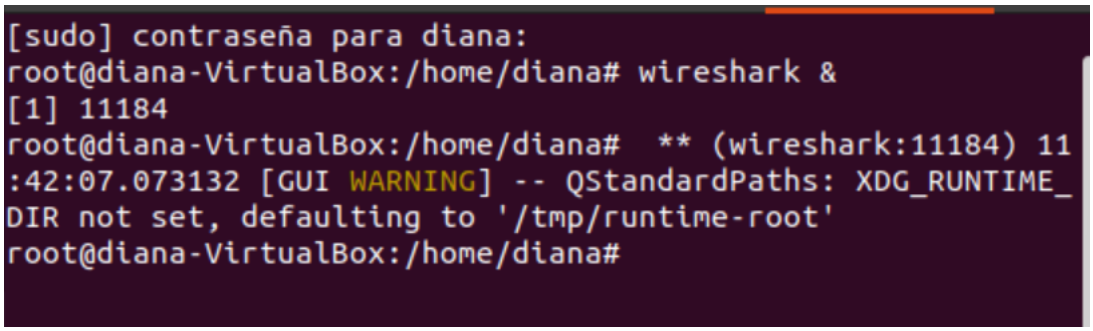
sudo apt update

sudo apt install wireshark

Durante la instalación, se te pedirá que configures los permisos para capturar paquetes sin privilegios de superusuario. Para ello, selecciona "Yes" y luego pulsa "Enter".

Una vez completado la instalación, podras ejecutar wireshark sin problema, recuerda ingresar en superusuario para hacerlo, mediante el siguiente comando:

sudo wireshark &



Anexo 2.- Pruebas de manejo de Mininet

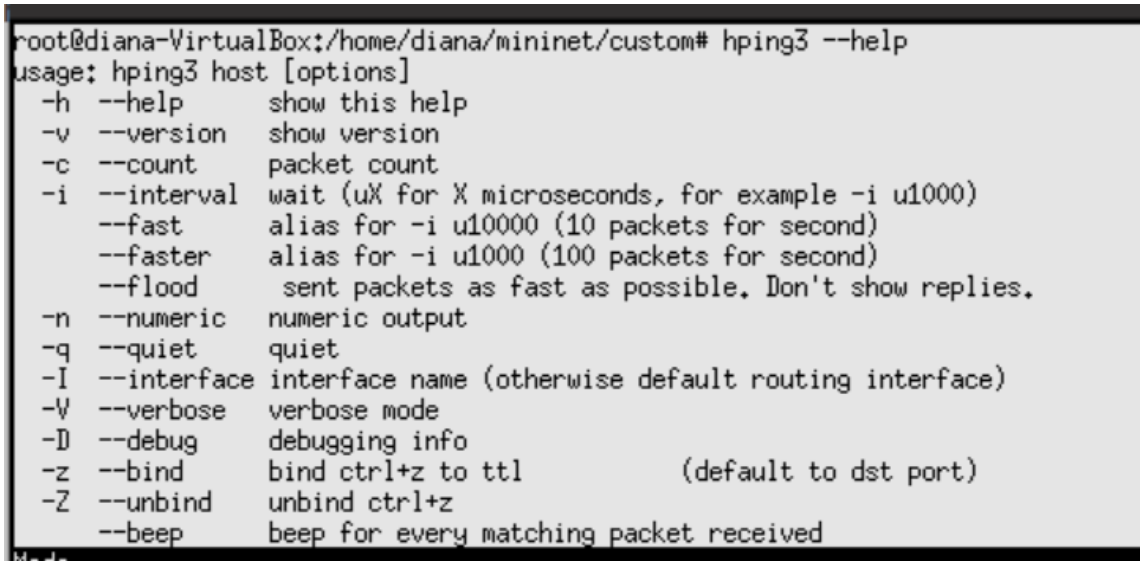
```
mininet> nodes
available nodes are:
c0 h1 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h2 h20 h21 h22 h23 h24 h25 h26 h27
 h28 h29 h3 h30 h31 h32 h4 h40 h41 h42 h5 h6 h7 h8 h9 s1 s18 s2 s28 s3 s38 s4 s4
1 s42 s5 s50 s54 s6 s64 s7 s74 s8
```

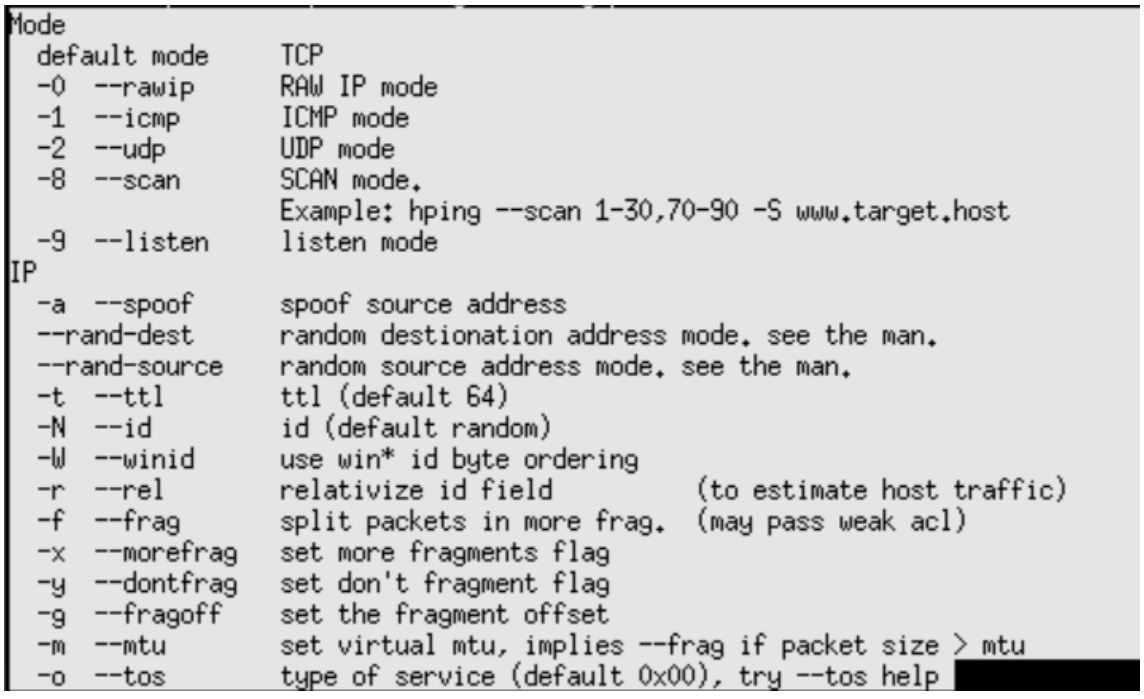
```
root@diana-VirtualBox: /ho...   root@diana-VirtualBox: /ho...   root@diana-VirtualBox: /ho...
s74-eth1<->s42-eth2 (OK OK)
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26
h27 h28 h29 h30 h31 h32 h40 h41 h42
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26
h27 h28 h29 h30 h31 h32 h40 h41 h42
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26
h27 h28 h29 h30 h31 h32 h40 h41 h42
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26
h27 h28 h29 h30 h31 h32 h40 h41 h42
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26
h27 h28 h29 h30 h31 h32 h40 h41 h42
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26
h27 h28 h29 h30 h31 h32 h40 h41 h42
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26
h27 h28 h29 h30 h31 h32 h40 h41 h42
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26
h27 h28 h29 h30 h31 h32 h40 h41 h42
```

```
root@diana-VirtualBox: /ho...   root@diana-VirtualBox: /ho...   root@diana-VirtualBox: /ho...
h26 h28 h29 h30 h31 h32 h40 h41 h42
h28 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25
h26 h27 h29 h30 h31 h32 h40 h41 h42
h29 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25
h26 h27 h28 h30 h31 h32 h40 h41 h42
h30 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25
h26 h27 h28 h29 h31 h32 h40 h41 h42
h31 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25
h26 h27 h28 h29 h30 h32 h40 h41 h42
h32 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25
h26 h27 h28 h29 h30 h31 h40 h41 h42
h40 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25
h26 h27 h28 h29 h30 h31 h32 h41 h42
h41 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25
h26 h27 h28 h29 h30 h31 h32 h40 h42
h42 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25
h26 h27 h28 h29 h30 h31 h32 h40 h41
*** Results: 0% dropped (1190/1190 received)
mininet>
```

```
mininet> h12 ping h1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=2.62 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.243 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.213 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.360 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=0.207 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=0.212 ms
^C
--- 10.0.0.1 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5219ms
rtt min/avg/max/mdev = 0.207/0.643/2.624/0.887 ms
mininet>
```

Anexo 3.- Comandos de Hping3 de red SDN





Anexo 4.- Pruebas de ping/Mininet

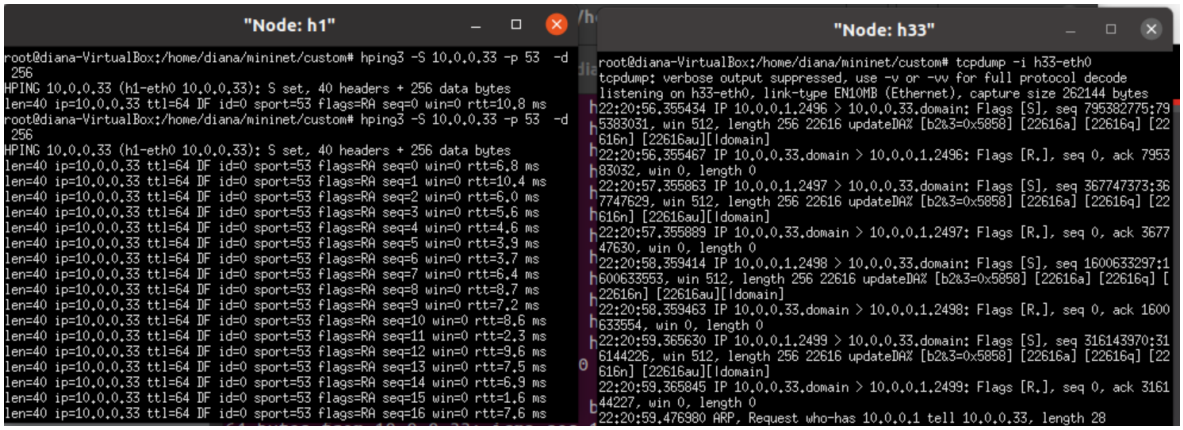
```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1.54 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.300 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=1.08 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.461 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.359 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.293 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.779 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.291 ms
^C
--- 10.0.0.2 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7118ms
rtt min/avg/max/mdev = 0.291/0.638/1.540/0.431 ms
mininet>
```

```
mininet> h3 ping 10.0.0.9
PING 10.0.0.9 (10.0.0.9) 56(84) bytes of data.
64 bytes from 10.0.0.9: icmp_seq=1 ttl=64 time=2.10 ms
64 bytes from 10.0.0.9: icmp_seq=2 ttl=64 time=0.356 ms
64 bytes from 10.0.0.9: icmp_seq=3 ttl=64 time=0.268 ms
64 bytes from 10.0.0.9: icmp_seq=4 ttl=64 time=0.270 ms
64 bytes from 10.0.0.9: icmp_seq=5 ttl=64 time=0.275 ms
64 bytes from 10.0.0.9: icmp_seq=6 ttl=64 time=0.236 ms
^C
--- 10.0.0.9 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5096ms
rtt min/avg/max/mdev = 0.236/0.583/2.096/0.677 ms
mininet>
```

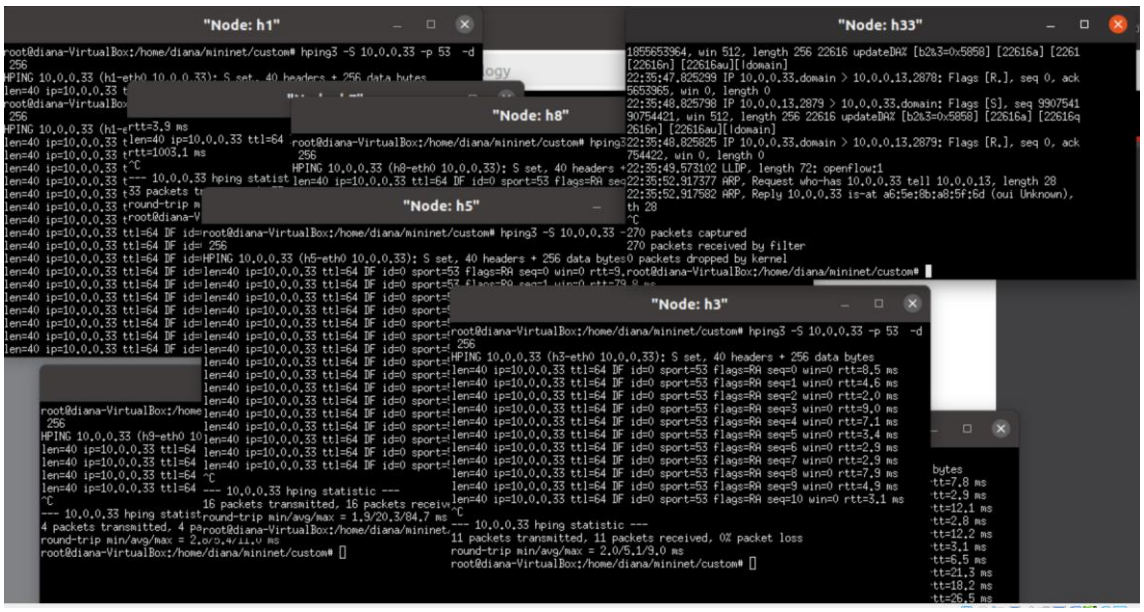
```
mininet> h5 ping 10.0.0.10
PING 10.0.0.10 (10.0.0.10) 56(84) bytes of data.
64 bytes from 10.0.0.10: icmp_seq=1 ttl=64 time=2.09 ms
64 bytes from 10.0.0.10: icmp_seq=2 ttl=64 time=0.319 ms
64 bytes from 10.0.0.10: icmp_seq=3 ttl=64 time=0.291 ms
64 bytes from 10.0.0.10: icmp_seq=4 ttl=64 time=0.326 ms
64 bytes from 10.0.0.10: icmp_seq=5 ttl=64 time=0.307 ms
64 bytes from 10.0.0.10: icmp_seq=6 ttl=64 time=0.312 ms
^C
--- 10.0.0.10 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5094ms
rtt min/avg/max/mdev = 0.291/0.607/2.089/0.662 ms
mininet>
```

Anexo 5.- Pruebas Hping3 Usuario/Servidor

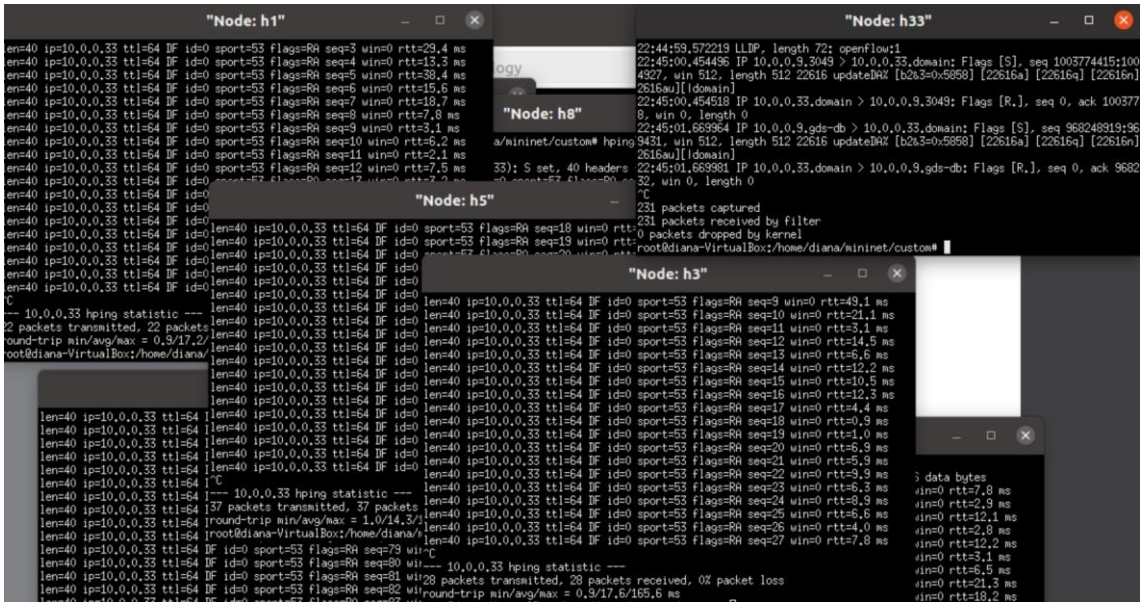
Anexo 6.- Usuario H1/ Puerto 53/ Longitud de Paquete 256 Bytes



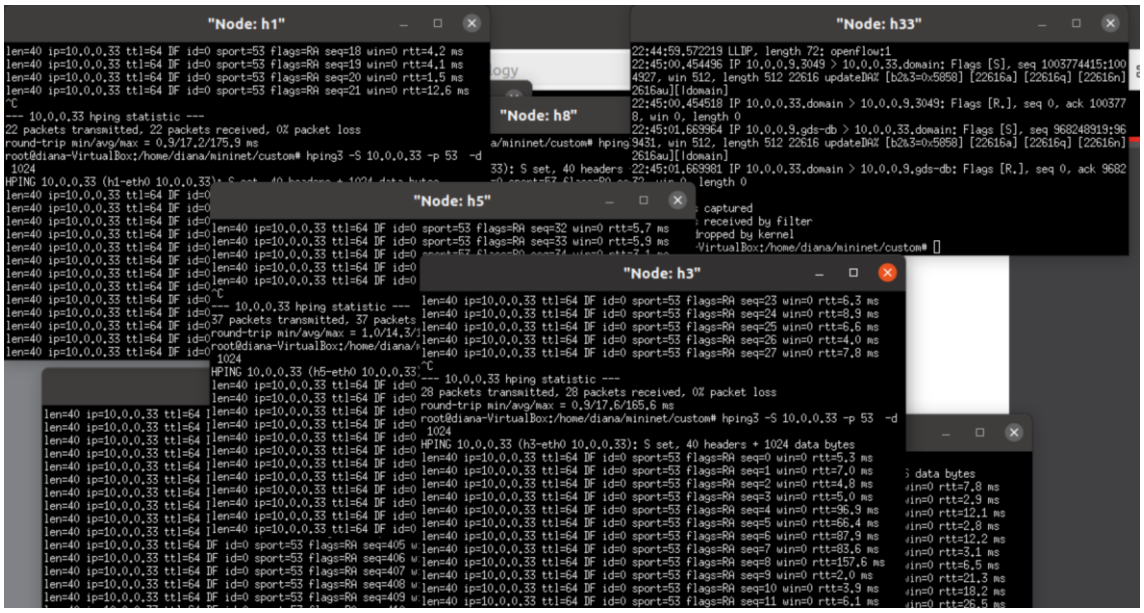
Anexo 7.- Usuarios simultáneos/ Puerto 53/Longitud de Paquete 256 Bytes



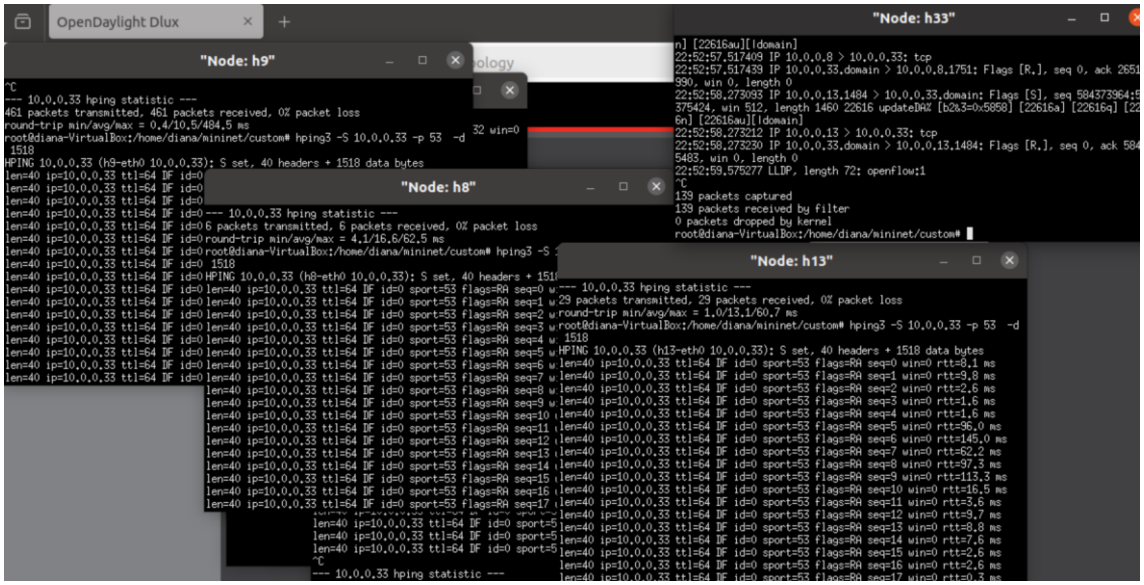
Anexo 8.- Usuarios simultáneos /Puerto 53 /Longitud de Paquete 512 Bytes



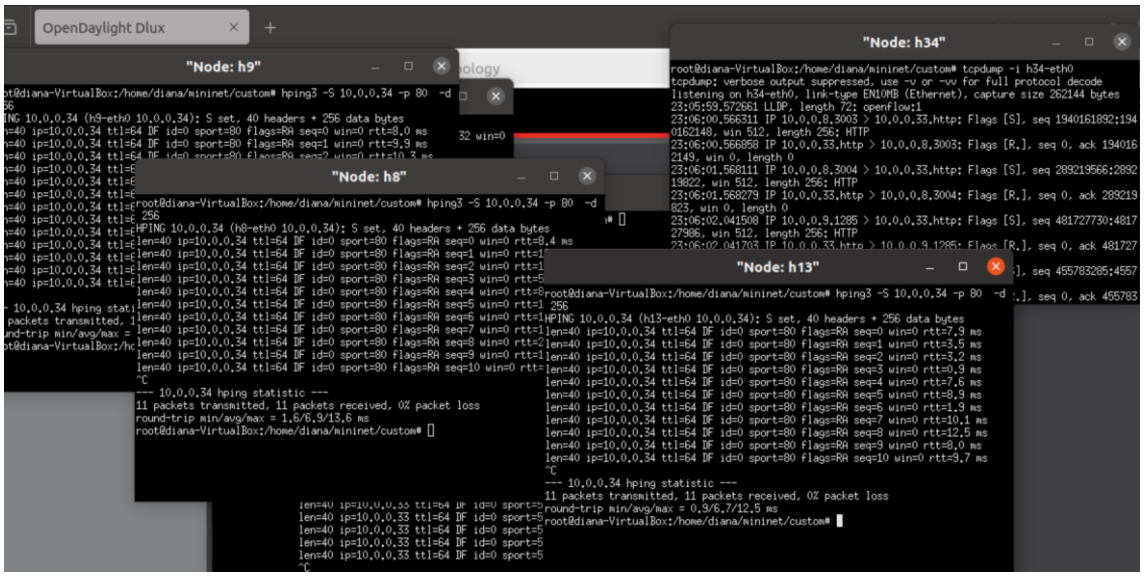
Anexo 9.- Usuarios simultáneos/Puerto 53/ Longitud de paquetes 1024



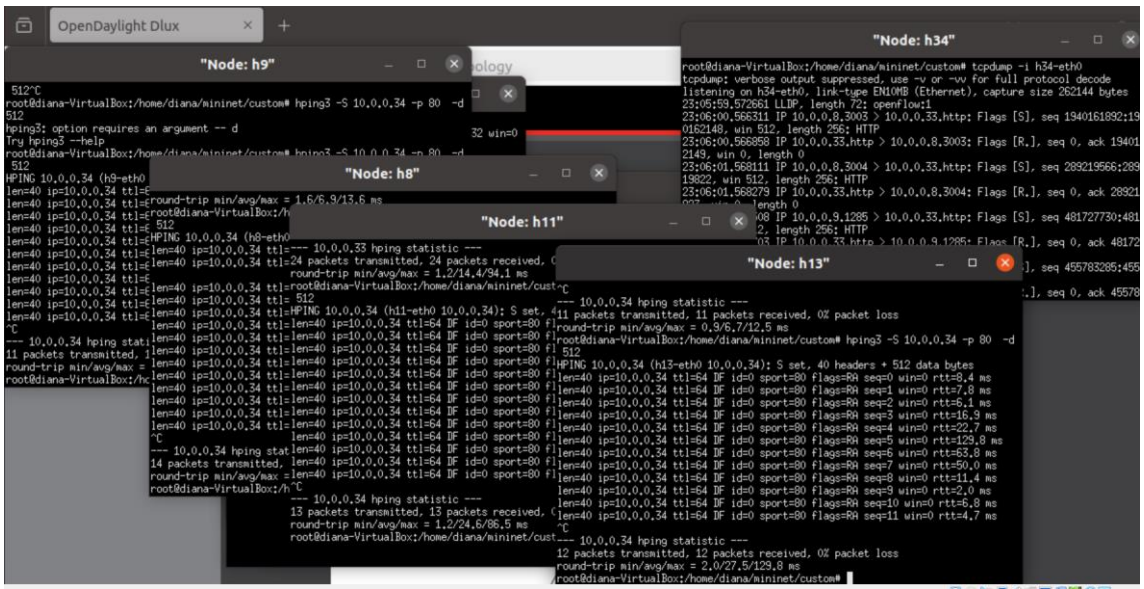
Anexo 10.- Usuarios Simultáneos/Puerto 53/Longitud de Paquete 1518



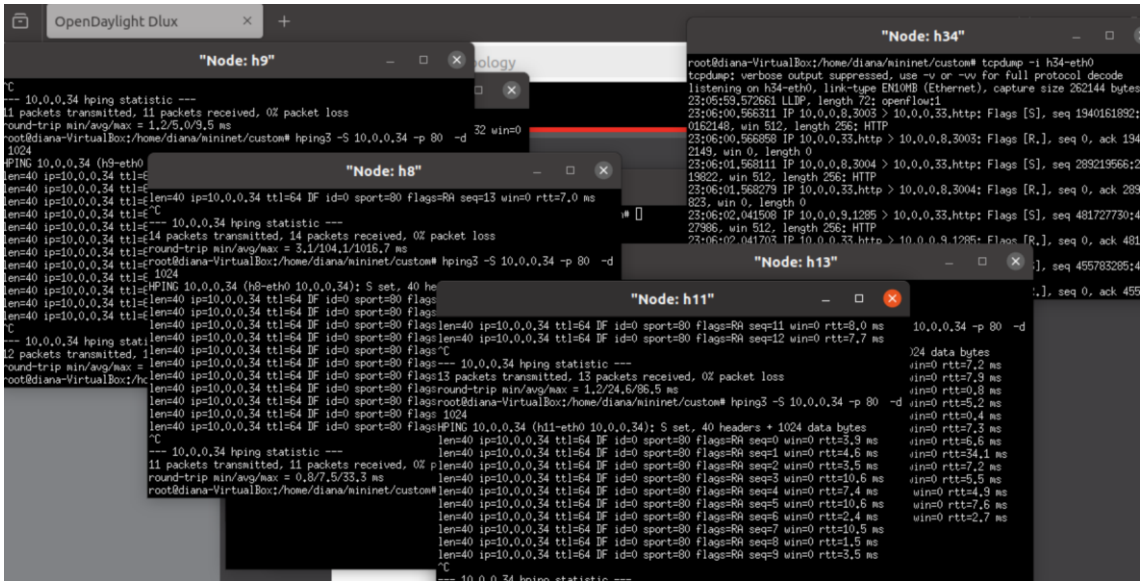
Anexo 11.- Usuarios Simultáneos/ Puerto 80/Longitud 256



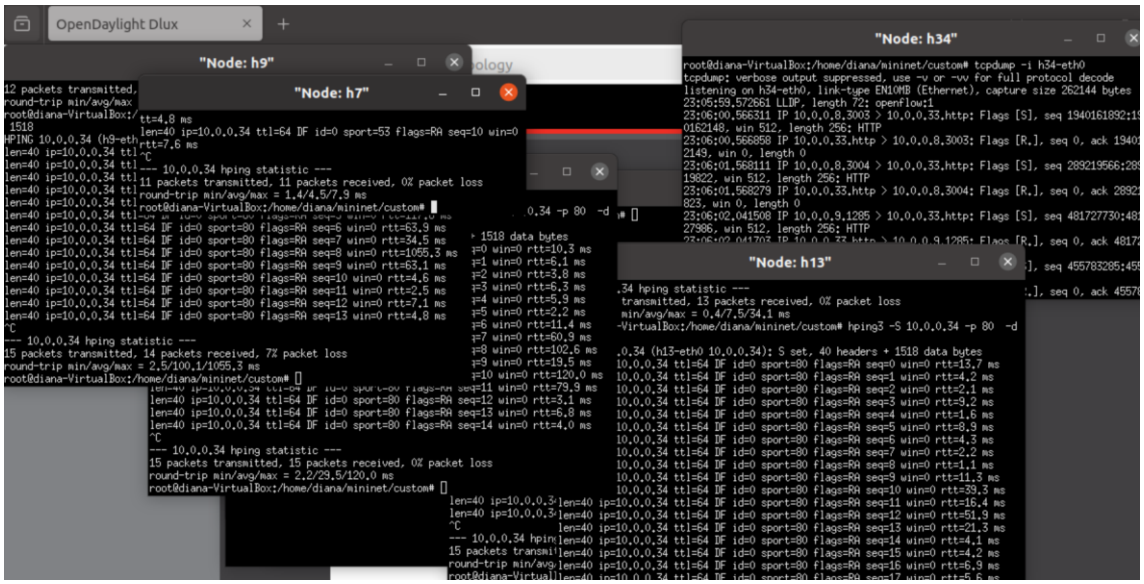
Anexo 12.- Usuarios Simultáneos/ Puerto 80/Longitud 512



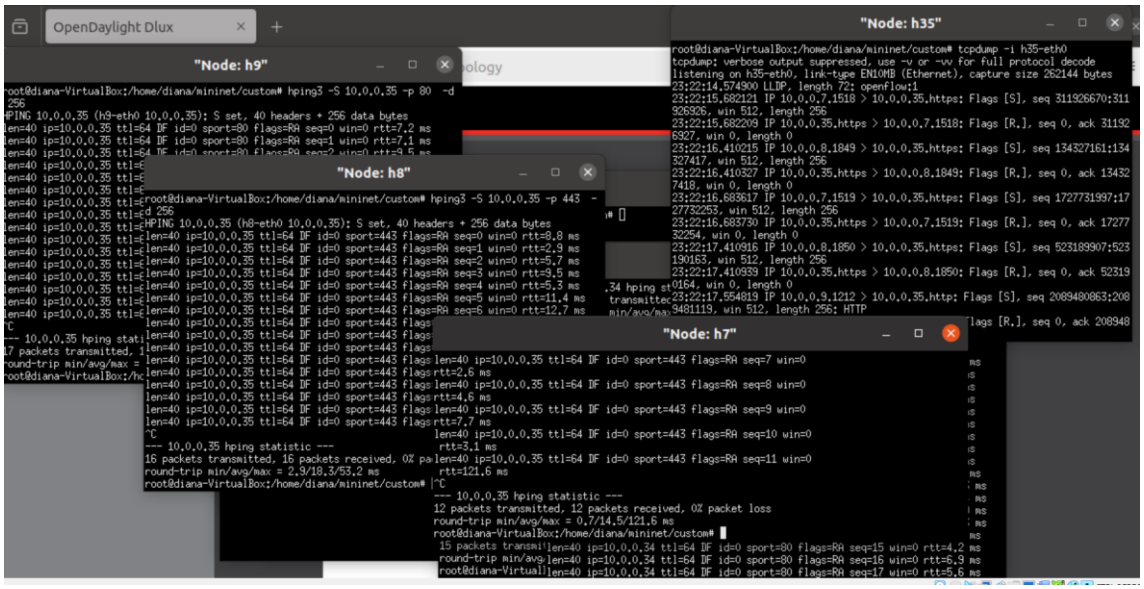
Anexo 13.- Usuarios Simultáneos/ Puerto 80/Longitud 1024



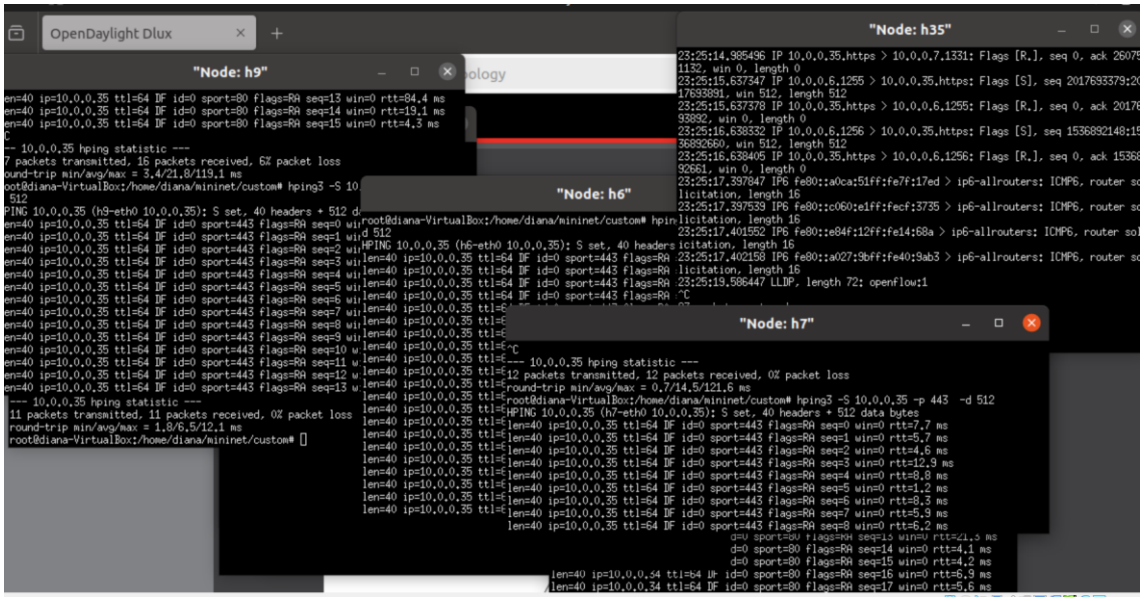
Anexo 14.- Usuarios Simultáneos/ Puerto 80/Longitud 1518



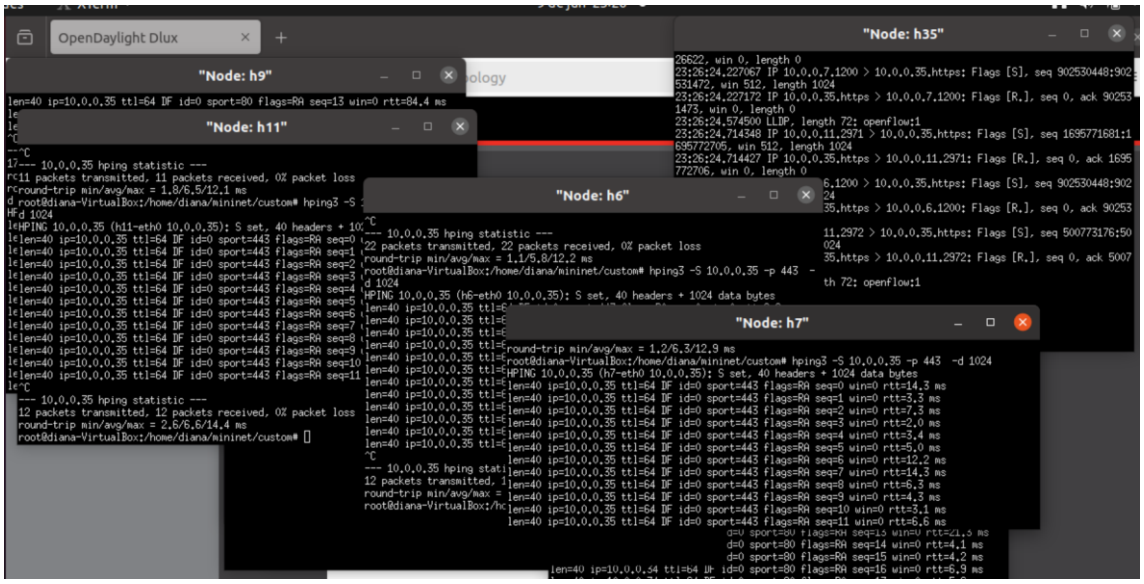
Anexo 15.- Usuarios simultáneos /Puerto 443/Longitud de Paquete 256



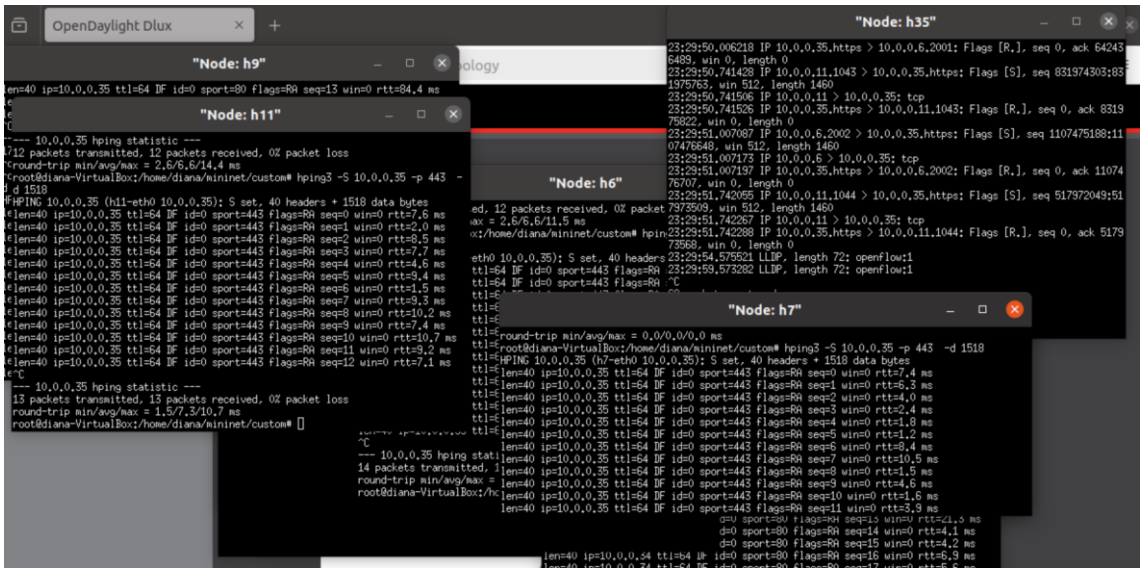
Anexo 16.- Usuarios simultáneos /Puerto 443/Longitud de Paquete 512



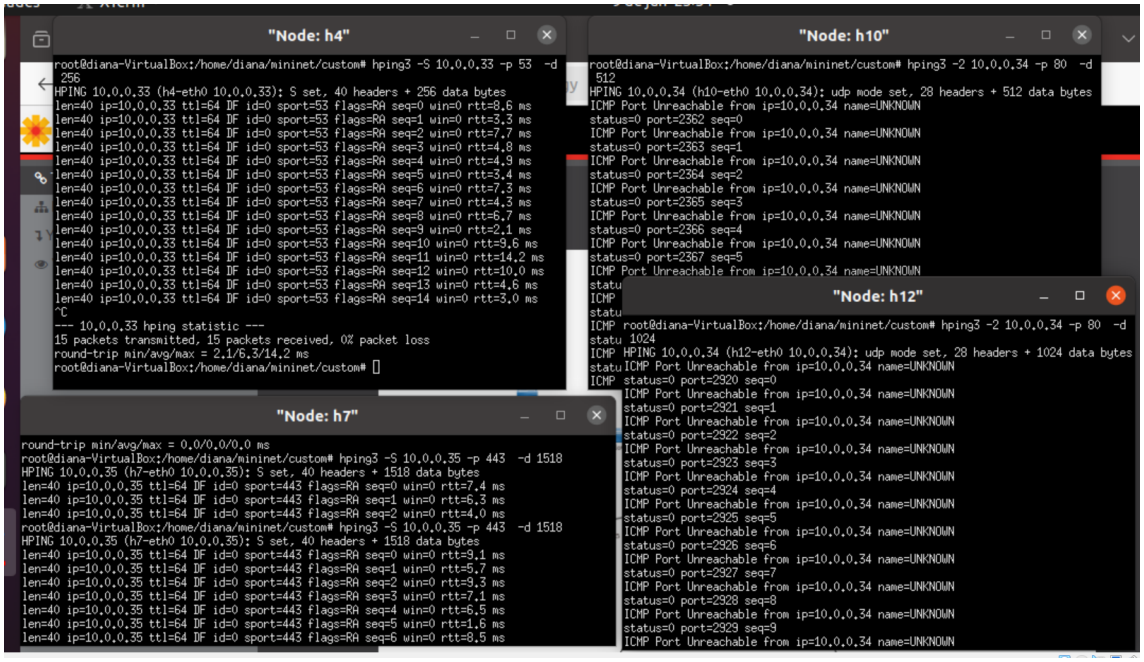
Anexo 17.- Usuarios simultáneos /Puerto 443/Longitud de Paquete 1024



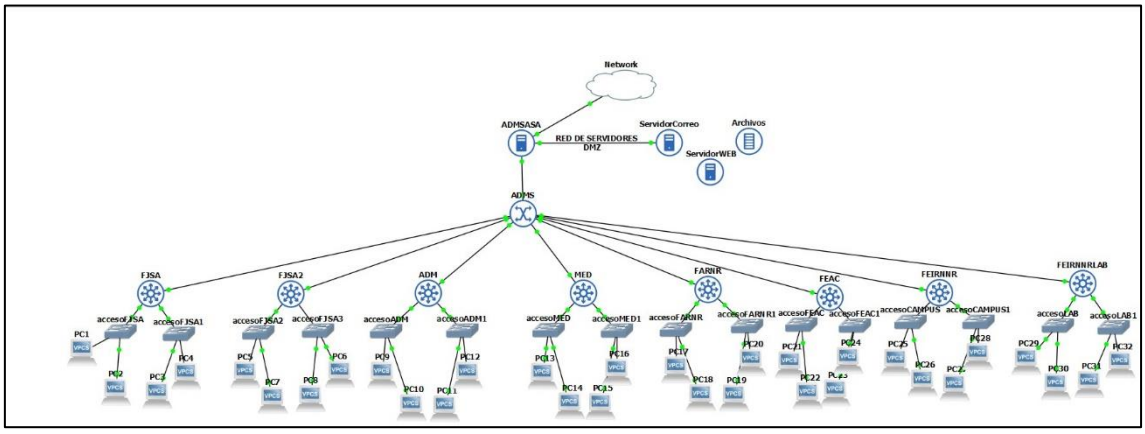
Anexo 18.- Usuarios simultáneos /Puerto 443/Longitud de Paquete 1518



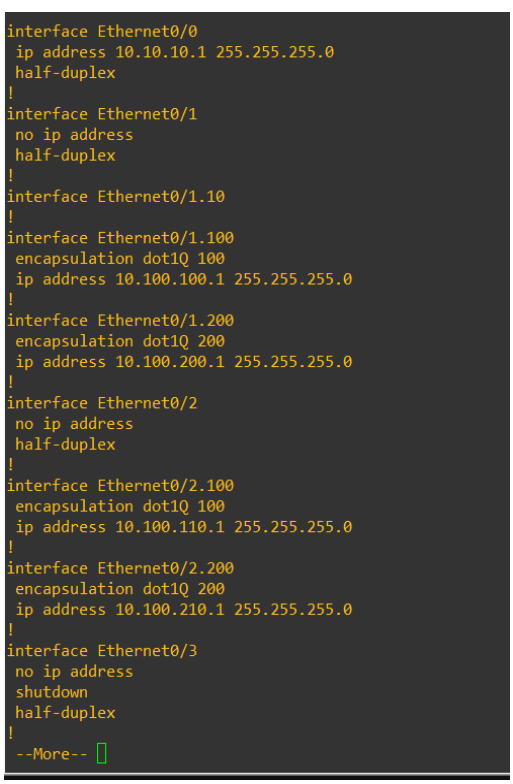
Anexo 19.- Usuarios Simultáneos /Puerto 53,80,443/Longitud de 256,512,1024,1518 Bytes



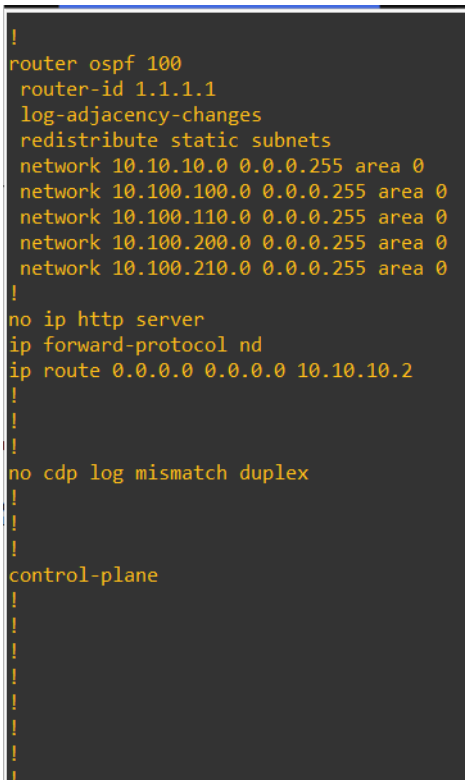
Anexo 20.- Topología de red en Software GNS3



Anexo 21.- Configuración Switch Capa 3

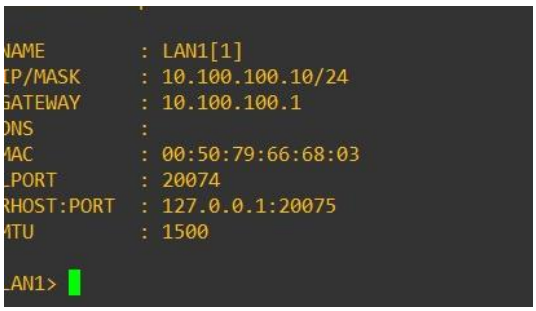


```
interface Ethernet0/0
 ip address 10.10.10.1 255.255.255.0
 half-duplex
!
interface Ethernet0/1
 no ip address
 half-duplex
!
interface Ethernet0/1.10
!
interface Ethernet0/1.100
 encapsulation dot1Q 100
 ip address 10.100.100.1 255.255.255.0
!
interface Ethernet0/1.200
 encapsulation dot1Q 200
 ip address 10.100.200.1 255.255.255.0
!
interface Ethernet0/2
 no ip address
 half-duplex
!
interface Ethernet0/2.100
 encapsulation dot1Q 100
 ip address 10.100.110.1 255.255.255.0
!
interface Ethernet0/2.200
 encapsulation dot1Q 200
 ip address 10.100.210.1 255.255.255.0
!
interface Ethernet0/3
 no ip address
 shutdown
 half-duplex
!
 --More--
```



```
!
router ospf 100
 router-id 1.1.1.1
 log-adjacency-changes
 redistribute static subnets
 network 10.10.10.0 0.0.0.255 area 0
 network 10.100.100.0 0.0.0.255 area 0
 network 10.100.110.0 0.0.0.255 area 0
 network 10.100.200.0 0.0.0.255 area 0
 network 10.100.210.0 0.0.0.255 area 0
!
no ip http server
ip forward-protocol nd
ip route 0.0.0.0 0.0.0.0 10.10.10.2
!
!
!
no cdp log mismatch duplex
!
!
!
control-plane
!
!
!
!
!
!
!
!
```

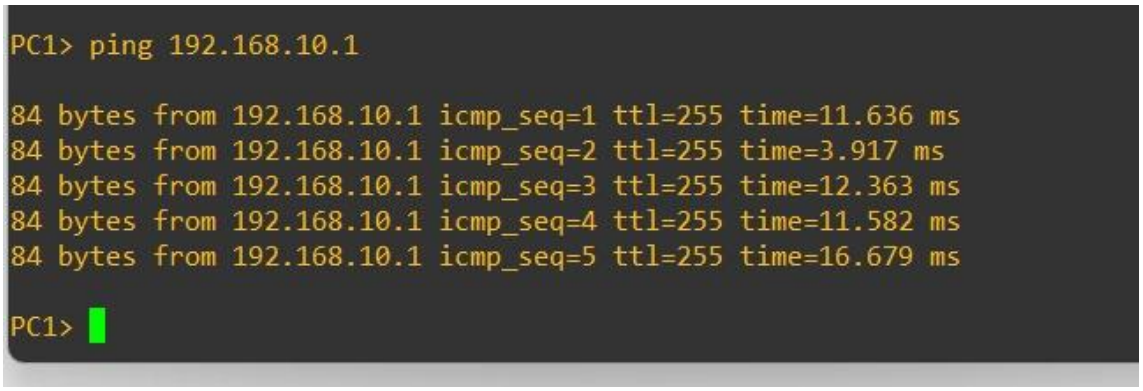
Anexo 22.- Configuración IP Usuario



```
NAME        : LAN1[1]
IP/MASK     : 10.100.100.10/24
GATEWAY     : 10.100.100.1
DNS         :
MAC         : 00:50:79:66:68:03
LPORT       : 20074
RHOST:PORT  : 127.0.0.1:20075
MTU         : 1500

LAN1>
```

Anexo 23.- Ping de Prueba /GNS3



Anexo 24.- Configuración Firewall ASA

```
ASA
interface GigabitEthernet0/0
 nameif inside
 security-level 100
 ip address 192.168.10.2 255.255.255.0
!
interface GigabitEthernet0/1
 nameif outside
 security-level 50
 ip address 172.168.10.2 255.255.255.0
!
ftp mode passive
object network inside-static
 subnet 192.168.10.0 255.255.255.0
object network LAN_INSIDE
 subnet 192.10.10.0 255.255.255.0
object network LAN2_INSIDE
 subnet 192.30.30.0 255.255.255.0
object network LAN3_vlan100
 subnet 10.100.100.0 255.255.255.0
access-list 100 extended permit icmp any any


object network LAN_INSIDE
 nat (inside,outside) dynamic interface
object network LAN2_INSIDE
 nat (inside,outside) dynamic interface
object network LAN3_vlan100
 nat (inside,outside) dynamic interface
access-group 100 in interface outside
route outside 0.0.0.0 0.0.0.0 172.168.10.1 1
route inside 10.100.100.0 255.255.255.0 192.168.10.1 1
route inside 100.100.100.100 255.255.255.255 192.168.10.1 1
route inside 192.10.10.0 255.255.255.0 192.168.10.1 1
route inside 192.30.30.0 255.255.255.0 192.168.10.1 1
```

Anexo 25.- Script de Python (CODIGO)

```
!/usr/bin/python
"""Topologia red Proyecto UNL ---
---Red Telecomunicaciones - Sede Central---
---Core---
---Distribucion Nodos---
---Acceso---
"""
from mininet.topo import Topo
from mininet.net import Mininet

class RedUNL(Topo):
    def build(self):
        # host Red CampusFJSA
        h1 = self.addHost('h1')
        h2 = self.addHost('h2')
        h3 = self.addHost('h3')
        h4 = self.addHost('h4')
        h5 = self.addHost('h5')
        h6 = self.addHost('h6')
        h7 = self.addHost('h7')
        h8 = self.addHost('h8')
        h9 = self.addHost('h9')
        h10 = self.addHost('h10')
        h11 = self.addHost('h11')
        h12 = self.addHost('h12')
        h13 = self.addHost('h13')
        h14 = self.addHost('h14')
        h15 = self.addHost('h15')
        h16 = self.addHost('h16')
        h17 = self.addHost('h17')
        h18 = self.addHost('h18')
        h19 = self.addHost('h19')
        h20 = self.addHost('h20')
        h21 = self.addHost('h21')
        h22 = self.addHost('h22')
        h23 = self.addHost('h23')
        h24 = self.addHost('h24')
```

```
h25 = self.addHost('h25')
h26 = self.addHost('h26')
h27 = self.addHost('h27')
h28 = self.addHost('h28')
h29 = self.addHost('h29')
h30 = self.addHost('h30')
h31 = self.addHost('h31')
h32 = self.addHost('h32')
h33 = self.addHost('h33')
h34 = self.addHost('h34')
h35 = self.addHost('h35')
h40 = self.addHost('h40')
h41 = self.addHost('h41')
h42 = self.addHost('h41')


# switch Core
s1 = self.addSwitch('s1')

# switch nodos distribucion
s2 = self.addSwitch('s2')   #FJSA
s3 = self.addSwitch('s3')   #FJSA2
s4 = self.addSwitch('s4')   #FARNR
s5 = self.addSwitch('s5')   #ADM
s6 = self.addSwitch('s6')   #UED
s7 = self.addSwitch('s7')    #LAB-FEIRNNR
s41 = self.addSwitch('s41') #FEIRNNR
s42 = self.addSwitch('s42') #FEAC

# switch nodos accesso FJSA
s8 = self.addSwitch('s8')


# switch nodos accesso FJSA2
s18 = self.addSwitch('s18')

# switch nodos accesso FARNR
s28 = self.addSwitch('s28')
```

```
# switch nodos accesso ADM
s38 = self.addSwitch('s38')

# switch nodos accesso UED
s50 = self.addSwitch('s50')

# switch nodos accesso LAB-FEIRNNR
s54 = self.addSwitch('s54')

# switch nodos accesso FEIRNNR
s64 = self.addSwitch('s64')

# switch nodos accesso FEAC
s74 = self.addSwitch('s74')

# enlaces distribución a core1
self.addLink(s2, s1)
self.addLink(s3, s1)
self.addLink(s4, s1)
self.addLink(s5, s1)
self.addLink(s7, s1)
self.addLink(s6, s1)
self.addLink(s41, s1)
self.addLink(s42, s1)

# enlace distribución as2
self.addLink(s8, s2)


 # enlace distribución FJSA2
self.addLink(s18, s3)

 # enlace distribución FARNR
self.addLink(s28, s4)

# enlace distribución ADM
self.addLink(s38, s5)

# enlace distribución UED
self.addLink(s50, s6)
```

```
# enlace distribución LAB-FEIRNNR
self.addLink(s54, s7)

# enlace distribución FEIRNNR
self.addLink(s64, s41)

# enlace distribución FEAC
self.addLink(s74, s42)

# enlace host acceso
self.addLink(h1,  s8)
self.addLink(h2,  s8)
self.addLink(h3,  s8)
self.addLink(h4,  s8)
self.addLink(h5,  s18)
self.addLink(h6,  s18)
self.addLink(h7,  s18)
self.addLink(h8,  s18)
self.addLink(h9,  s28)
self.addLink(h10,  s28)
self.addLink(h11,  s28)
self.addLink(h12,  s28)
self.addLink(h13,  s38)
self.addLink(h14,  s38)
self.addLink(h15,  s38)
self.addLink(h16,  s38)
self.addLink(h17,  s50)
self.addLink(h18,  s50)
self.addLink(h19,  s50)
self.addLink(h20,  s50)
self.addLink(h21,  s54)
self.addLink(h22,  s54)
self.addLink(h23,  s54)
self.addLink(h24,  s54)
self.addLink(h25,  s64)
self.addLink(h26,  s64)
self.addLink(h27,  s64)
self.addLink(h28,  s64)
self.addLink(h29,  s74)
```

```
        self.addLink(h30,  s74)
        self.addLink(h31,  s74)
        self.addLink(h32,  s74)
        self.addLink(h33,  s1)
        self.addLink(h34,  s1)
        self.addLink(h35,  s1)

 # Permite al archivo ser importado con `mn --custom  <filename> --topo`
topos = {'RedUNL': (lambda: RedUNL())}
net = Mininet(topo=RedUNL())
net.start()
net.pingAll()
net.stop()
```

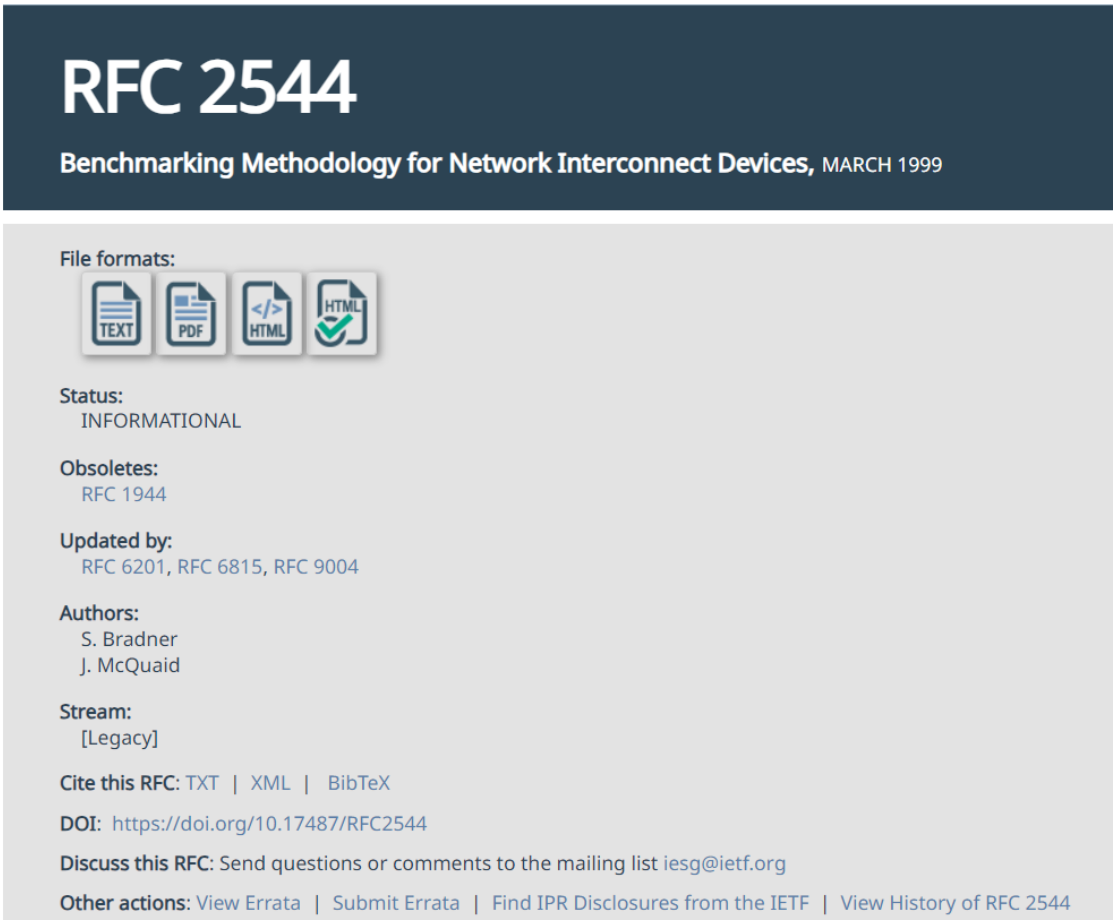
Anexo 26.- SERIE G1010: SISTEMAS Y MEDIOS DE TRANSMISIÓN

Cuadro I.2/G.1010 Objetivos de calidad de funcionamiento para aplicaciones datos

Medio	Aplicación	Grado de simetría	Velocidades de datos típicas	Parámetros clave y valores de objetivo para la calidad de funcionamiento		
				Tiempo de transmisión en un sentido (Nota)	Variación de retardos	Pérdida de información
Datos	Navegación en la web – HTML	Principalmente un sentido	~10 KB	Preferido < 2 s/página Aceptable < 4 s/página	N.A.	Nula
Datos	Transferencia/ recuperación de gran volumen de datos	Principalmente un sentido	10 KB-10 MB	Preferido < 15 s Aceptable < 60 s	N.A.	Nula
Datos	Servicios de transacciones de alta prioridad, como comercio electrónico, ATM	Dos sentidos	< 10 KB	Preferido < 2 s Aceptable < 4 s	N.A.	Nula
Datos	Medio dirigido/control	Dos sentidos	~ 1 KB	< 250 ms	N.A.	Nula
Datos	Imagen fija	Un sentido	< 100 KB	Preferido < 15 s Aceptable < 60 s	N.A.	Nula
Datos	Juegos interactivos	Dos sentidos	< 1 KB	< 200 ms	N.A.	Nula
Datos	Telnet	Dos sentidos (asimétrico)	< 1 KB	< 200 ms	N.A.	Nula
Datos	Correo electrónico (acceso a servidor)	Principalmente un sentido	< 10 KB	Preferido < 2 s Aceptable < 4 s	N.A.	Nula
Datos	Correo electrónico (transferencia de servidor a servidor)	Principalmente un sentido	< 10 KB	Puede ser varios minutos	N.A.	Nula
Datos	Fax ("tiempo real")	Principalmente un sentido	~ 10 KB	< 30 s/página	N.A.	$<10^{-6}$ BER
Datos	Fax (almacenamiento y retransmisión)	Principalmente un sentido	~ 10 KB	Pueden ser varios minutos	N.A.	$<10^{-6}$ BER
Datos	Transacciones de baja prioridad	Principalmente un sentido	< 10 KB	< 30 s	N.A.	Nula
Datos	Usenet	Principalmente un sentido	Puede ser 1 MB o más	Pueden ser varios minutos	N.A.	Nula
NOTA – En algunos casos, puede ser más apropiado considerar estos valores como tiempos de respuesta.						

Para mayor Información:
https://www.itu.int/rec/dologin_pub.asp?lang=s&id=T-REC-G.1010-200111-I!!PDF-S&type=items

Anexo 27.- RFC 2544: Metodología del rendimiento de dispositivos de red.



RFC 2544

Benchmarking Methodology for Network Interconnect Devices, MARCH 1999

File formats:

TEXT PDF HTML HTML

Status:
INFORMATIONAL

Obsoletes:
RFC 1944

Updated by:
RFC 6201, RFC 6815, RFC 9004

Authors:
S. Bradner
J. McQuaid

Stream:
[Legacy]

Cite this RFC: TXT | XML | BibTeX

DOI: https://doi.org/10.17487/RFC2544

Discuss this RFC: Send questions or comments to the mailing list iesg@ietf.org

Other actions: View Errata | Submit Errata | Find IPR Disclosures from the IETF | View History of RFC 2544

Para mayor información: https://datatracker.ietf.org/doc/html/rfc2544

Anexo 28.- Traducción Abstract

FINE- NED ENGLISH
tßßOUßO£ IßfTIT¥T£
Líderes en la Enseñanza del Inglés

Ing. María Belén Novillo Sánchez.
ENGLISB TEACHER - FINE TUNED ENGLISH CIA **LTDA.**

CERTIFICA:

Que el documento aquí compuesto es fiel traducción del idioma español a1 idioma inglés del trabajo de titulación **"Desarrollo de una propuesta de diseño basada en redes definidas por software (SDN) para la red de datos de la Universidad Nacional Loja, campus la Argelia, ciudad de Loja"** autoría de Diana Marisol Lozano Lozano, con número de cédula 1950010247, estudiante de la carrera de Ingeniería en Telecomunicaciones de la Universidad Nacional de Loja.

Lo certifico en honor a la verdad y autorizo a la interesada hacer uso del presente en lo que a sus intereses convenga.

Loja, 4 de junio del 2024

FINE - TUNED ENGLISH - CIA. LTDA. - LOJA - ECUADOR

Ing. María Belén Novillo Sánchez.
ENGLISH **TEACHER- FINE TUNED ENGLISH CIA LTDA.**