UNIVERSIDAD SEÑOR DE SIPÁN

FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO

ESCUELA ACADÉMICA PROFESIONAL DE INGENIERÍA DE SISTEMAS

TESIS

ANÁLISIS COMPARATIVO DEL RENDIMIENTO DE CONTROLADORES DE REDES DEFINIDAS POR SOFTWARE DE TIPO OPEN SOURCE PARA LA GESTIÓN DE UNA RED DE COMPUTADORAS

PARA OPTAR TÍTULO PROFESIONAL DE INGENIERO DE SISTEMAS

Autor(a):
Bach. Santisteban Ynga, Bicry Raúl
https://orcid.org/0000-0002-6498-0752

Asesor:
Ing. Mejía Cabrera, Iván
https://orcid.org/0000-0002-0007-0928

Línea de Investigación:
Infraestructura, Tecnología y Medio Ambiente

Pimentel – Perú
2021

APROBACIÓN DEL JURADO

ANÁLISIS COMPARATIVO DEL RENDIMIENTO DE CONTROLADORES DE REDES DEFINIDAS POR SOFTWARE DE TIPO OPEN SOURCE PARA LA GESTIÓN DE UNA RED DE COMPUTADORAS

_____

Bach. Santisteban Ynga, Bicry Raúl

Autor

_____

Ing. Mejía Cabrera, Iván

Asesor

_____ _____

Grado, apellidos y nombres
Presidente de Jurado

Grado, apellidos y nombres
Secretario de Jurado

_____

Grado, apellidos y nombres

Vocal de Jurado

DEDICATORIA

Para las personas más importantes y especiales en mi vida: mi adorada madre, amadísimas esposa e hijas, quienes, a diario con sus muestras de cariño, apoyo incondicional, paciencia y comprensión, se han convertido en la fortaleza de mi vida, la motivación más hermosa que puede tener un hombre para salir adelante.

¡Miren! Los hijos son una herencia de parte de Jehová; el fruto del vientre es un galardón".
Salmos 127:3

AGRADECIMIENTO

“Confía en Jehová con todo tu corazón
y no te apoyes en tu propio entendimiento.
Tómalo en cuenta en todos tus caminos,
y él hará rectas tus sendas.”.
Proverbios 3:5-6

.
Agradezco en primer lugar a mi Padre Celestial, Jehová Dios, pues si no fuera por su voluntad, nada de esto se habría logrado.

También agradezco de manera muy especial a mi compañera de vida, mi esposa; su constante apoyo, ánimo y paciencia, es fundamental para lograr todo lo que nos propongamos.

A mi asesor, por su respaldo en el proyecto, su conocimiento, experiencia, tiempo y dedicación garantizaron el poder cumplir con los objetivos trazados en mi carrera profesional.

Espero sinceramente ser reciproco con cada uno de las personas que han sumado para llegar a la meta.

RESUMEN

La evolución en la era digital trae consigo innovadores proyectos en el terreno de las Tecnologías de la Información y Comunicación, entre las cuales destacan las herramientas de gestión.

Durante los últimos años está en auge una herramienta tecnológica que presenta como fortalezas su dinámica, escalable, adaptable y rentable forma de administrar las redes de datos, centralizándola y aislando las funciones de envío de la red de las funciones de control, convirtiéndola en gestionable.

Las Redes Definidas por Software (SDN), la constituyen sus tres capas: Infraestructura, Control y Aplicaciones, esta investigación tiene su alcance específicamente en la médula o esencia de la arquitectura, conocida como controlador, el encargado de gestionar la inteligencia de la red.

Estudios previos de la Open Networking Foundation (ONF) nos muestran las características, funcionamiento y elementos para la selección del Controlador SDN. La presente investigación tiene como objetivo el despliegue en un entorno simulado de una Arquitectura SDN, utilizando las versiones actualizadas y herramientas de prueba basado en los indicadores de rendimiento y latencia. Como aporte está el estudio comparativo de controladores OpenSource que permitan a los profesionales del mundo de las redes obtener una guía confiable para futuros proyectos de migración SDN.

Los resultados, basados en pruebas de configuración y desempeño evidencian que el controlador SDN OpenDaylight otorga un óptimo rendimiento.

Palabras Clave – Redes definida por software, SDN, controlador, capa de control, capa de datos.

ABSTRACT

The evolution in the digital age brings with it innovative projects in the field of Information and Communication Technologies, among which management tools stand out.

During the last few years, a technological tool has been on the rise that presents as strengths its dynamic, scalable, adaptable and profitable way of managing data networks, centralizing it and isolating the sending functions of the network from the control functions, making it manageable.

Software Defined Networks (SDN) are made up of its three layers: Infrastructure, Control and Applications, this research has its scope specifically in the core or essence of the architecture, known as the controller, in charge of managing the intelligence of the network.

Previous studies by the Open Networking Foundation (ONF) show us the characteristics, operation and elements for the selection of the SDN Controller. The objective of this research is to deploy an SDN Architecture in a simulated environment, using updated versions and testing tools based on performance and latency indicators. As a contribution there is the comparative study of OpenSource controllers that allow professionals in the world of networks to obtain a reliable guide for future SDN migration projects.

The results, based on configuration and performance tests, show that the SDN OpenDaylight controller provides optimal performance.

Keywords - Software-defined networks, SDN, controller, control layer, data layer.

TABLA DE CONTENIDO

TABLA DE FIGURAS

ÍNDICE DE TABLAS

I. INTRODUCCION

1.1. Realidad Problemática

En nuestra era digital, la tecnología evidencia un veloz y exponencial desarrollo, las novedosas ideas e importantes planes en Tecnologías de la Información y Comunicación permiten obtener un abanico de herramientas tecnológicas a la vanguardia. Esto a su vez deja en evidencia las altas deficiencias y retos de las redes convencionales en cuanto a satisfacer requerimientos que involucran niveles de complejidad, implementación y administración.

Open Networking Foundation (ONF,2012) especifica que los inconvenientes e incidencias más recurrentes de las redes tradicionales se basan en su arquitectura, aquella que resulta altamente compleja, con políticas rudas e inconsistentes, nula escalabilidad e incita al monopolio en los proveedores de equipos de comunicaciones, los cuales recomiendan instalar y configurar equipos de la misma marca en un proyecto.

Las empresas y usuarios están a la expectativa de obtener los mejores resultados en las adquisiciones de hardware, software y sus servicios que involucran, en el caso de las Redes y Telecomunicaciones no ha sido la excepción. Aunque ha demorado más tiempo de lo proyectado, las importantes empresas otorgan herramientas y recursos a nivel de usuario y Administrador de Red, que permiten obtener información concisa e integra. Asimismo, se busca el mejor rendimiento y es donde aparece como innovadora solución tecnológica las Redes Definidas por Software, con su enfoque programable, centralizado, que ofrece las mejores expectativas para afrontar los altos requerimientos que hoy adolecen las redes tradicionales.

Ante las expectativas generadas sobre el nuevo paradigma de las redes basado en un controlador o cerebro de administración, centralizada, este trabajo busca investigar y generar pruebas para evaluar los controladores SDN.

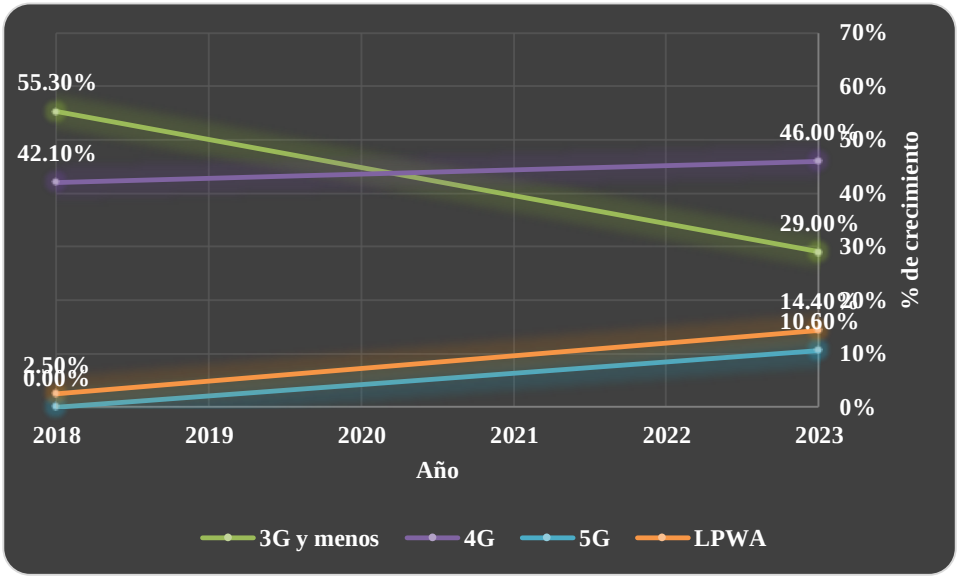

Figura 1. Porcentaje de Crecimiento en el uso de dispositivos móviles y conexiones. Fuente: Cisco Annual Internet Report, 2018–2023

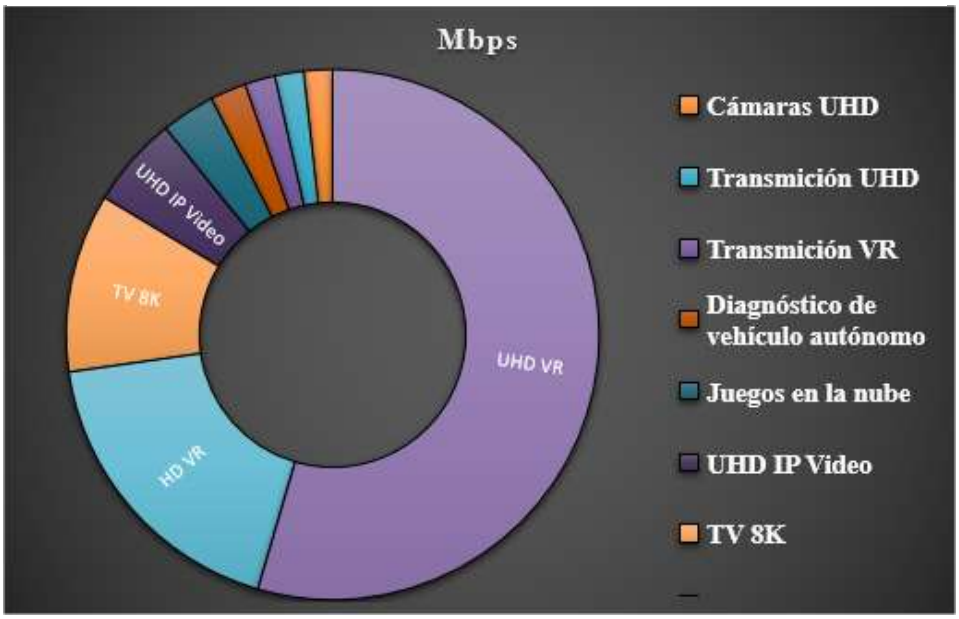

Figura 2. Consumo proyectado de ancho de banda por las diferentes aplicaciones. Fuente: Cisco Annual Internet Report, 2018–2023

Asimismo, nos encontramos ante el requerimiento de alta disponibilidad, conocida como “las cinco 9”, es decir un 99.999%, que equivale a un máximo de 5.26 minutos de inactividad al año, que como Administradores de red tenemos que garantizar, que presenta como incógnita ¿Qué herramientas tecnológicas me permitirían alcanzar el objetivo?

Según la Networking Foundation (ONF), define las SDN como una herramienta tecnológica emergente que ofrece dinamismo en su forma de gestionar las redes de datos, programabilidad centralizada en el cerebro de la arquitectura, conocida como controlador SDN. Lo interesante es la forma como desacopla las funciones de control del envío y la de paquetes de red.

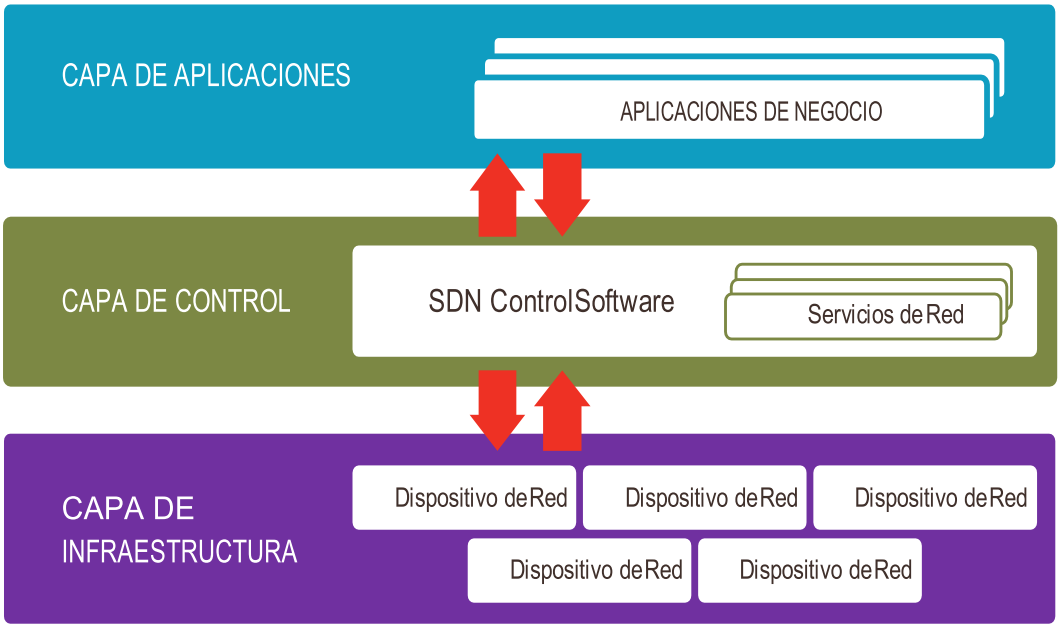

Figura 3. Arquitectura de una Red Definida por Software.
Fuente: Elaboración propia

Las SDN continúan creciendo en grado, diversidad y valor, por lo que deriva inevitable reflexión, comparación y medición de su arquitectura, lógica y componentes; que se implementen para futuras migraciones de tipo parcial o total de las redes de datos. Como innovador paradigma tiene la atención e interés de los investigadores y

profesionales del sector. La mayoría de los instrumentos que buscan reproducir la conducta real de una topología de red teniendo en cuenta la arquitectura SDN se basan en un entorno virtual o simulado, y tienen como objetivo generar resultados medibles, en base a pruebas de esfuerzo y funcionales que miden el desempeño de estas redes, y por ende desarrollar nuevas aplicaciones, usar protocolos y controladores en su versión actualizada para aprovechar las bondades que nos ofrecen a la vanguardia de la tecnología.

Para obtener una medición actualizada y notable es requerido estudiar las métricas y su desempeño bajo escenarios específicos y con mayor similitud a la realidad.

1.2. Antecedentes de Estudio.

Hasta la fecha, encontramos escasas investigaciones cuyo alcance sea la evaluación y comparación de controladores SDN (Redes Definidas por Software), en este trabajo se destacan las siguientes:

Rowshanrad, Abdi & Keshtgari, (2016) se enfoca en realizar un análisis comparativo orientado a la calidad de servicio, también conocida como QoS, en los controladores Floodlight y OpenDayLight. Su trabajo es importante debido a los altos requerimientos de los usuarios hacia las redes de datos, donde tenemos que tomar decisiones en base a parámetros de calidad de servicio, como el tiempo de retardo, la cantidad de perdida de paquetes, el tipo de topología, los niveles y su prioridad de carga de tráfico en los dispositivos.

Hoang & Pham, (2015) indica que a medida que SDN obtiene más implementaciones, los requisitos deben cerrar las brechas del conocimiento, y poder desplegarlos en sistemas de producción que requieren alta disponibilidad y seguridad. Se discute y expone los detalles de diseño de un controlador SDN, el énfasis está en las

interfaces, ya que son piezas importantes para la comunicación con los dispositivos y recursos de red.

Khattak, Awais & Iqbal, (2015) explica que a pesar de que encontramos estudios de evaluación y comparación de controladores, OpenDaylight no ha sido considerado o en su defecto descartado en base a criterios de cada autor. Aquellos que sugieren que OpenDaylight controlador SDN (ODL) no es lo suficientemente maduro para ser desplegado en la industria. Al denotar escasa confianza en los resultados sugiere utilizar el esquema ECHO, que utiliza modelos de cadena de la distribución Markov. La hipotética razón puede ser que el plugin OpenFlow en el modo rendimiento no otorga un aceptable funcionamiento. Por eso concluyen en que los resultados de evaluación comparativa con Cbench no es muy exitosa.

Laissaoui, Idboufker, Elassali & Baamrani, (2016) sostiene dos incógnitas para las SDN: ¿Con cuanta rapidez un controlador SDN puede responder a los mensajes de solicitud en la ruta de comunicación, y que capacidad de solicitudes puede atender en un tiempo determinado? Para ello es necesario un estudio comparativo de los controladores que utilizan el protocolo OpenFlow, entre ellos se destaca Floodlight, Beacon, Pox y Ryu, en base a métricas de latencia y rendimiento. Los principales resultados muestran que Floodlight es el mejor controlador para usar en una red que usa OpenFlow 1.0.

Yamei, Qing & Qi, (2016) indica que el controlador SDN es el componente más importante de la arquitectura. Por ello comparo OpenDaylight y ONOS en pruebas de rendimiento. ONOS podría llegar a 40k respuesta /s. Pero OpenDaylight solo tiene 0.3k de respuesta /s. Esto se debe principalmente al ambiente del experimento, debido a que se utilizó una PC con solo 2GB de memoria RAM. Se concluye que ONOS presenta mejor rendimiento en las características de interfaz GUI, clústeres, conexión, conmutación y rendimiento, mientras que su

oponente OpenDaylight presenta mejores resultados durante el descubrimiento de topología de red y estabilidad.

Salman, Elhajj, Kayssi & Chehab, (2016) reitera la importancia del controlador como componente principal de la arquitectura SDN, a su vez indica que, al encontrar diversidad de arquitecturas, urge la necesidad de una evaluación comparativa y no solo cuantitativa, teniendo específicos criterios de carácter cualitativos.

Bispo, Corujo & Aguiar, (2017) expresa que el crecimiento tecnológico está impulsando el desarrollo de soluciones SDN, aquellas que se distinguen en sus características y funciones. Lo que resalta es que las evaluaciones más recientes de SDN se enfocan solo en el rendimiento y no utilizan las últimas versiones que de forma constante son lanzadas al mercado. Por ello en esa comparación, se intentó no solo comparar controladores de distinta naturaleza, sino también controladores con similares características para evaluar sus diferencias de rendimiento.

Rajaratnam, Kadikar, Prince & Valarmathi, (2017) explica que la convergencia de las redes de datos con los servicios CLOUD deben interactuar para lograr el objetivo. La idea básica detrás de esta experimentación es promocionar el despliegue del controlador ONOS en las topologías existentes y futuras. Como resultado la topología en malla demuestra ser la mejor para redes de alta velocidad con latencia extremadamente bajas en comparación a otras topologías.

Aliyu, Bull & Abdallah, (2017) sostiene que la arquitectura tradicional de redes de datos es compleja, difícil de administrar, inflexible y estática. Al igual que las anteriores investigaciones obtuvieron resultados en base a las características del rendimiento de la red, delay y jitter.

Darianian, Williamson & Haque, (2017) afirma que hasta donde tenemos conocimiento, no hay un estudio reciente en la literatura centrada en la evaluación del rendimiento de ONOS y OpenDaylight. El estudio práctico concluye que, en la mayoría de las pruebas, ONOS y OpenDaylight presentan un mejor rendimiento y por ende una cifra menor de latencia. La configuración que interviene en los resultados, es que el Hyper-threading debe estar habilitado. Concluye indicando un extraño comportamiento de OpenDaylight en las pruebas de rendimiento, debido a un presunto error en su Plugin OpenFlow.

Kaur & Gupta, (2018) explica que las redes comunes son inflexibles, carecen de una administración descentralizada y los dispositivos de red que utiliza no son programables. En el mercado se ofrece distintos controladores OpenSource y propietarios o de pago. La diferencia más notable entre ellos es el lenguaje de programación en el que fueron programados y la compatibilidad con el estándar OpenFlow más utilizado hasta la fecha. La mayoría de los controladores están basados en Java o Python.

Mamushiane, Lysko & Dlamini, (2018) aborda una realidad diferente, aquella que escala a evaluar los controladores en ámbitos de gran envergadura y criticidad como las SD-WAN. Dos de las incógnitas más importantes son: ¿Con que rapidez un controlador puede responder a los mensajes PACKET_IN? (El conmutador envía las solicitudes PACKET_IN al controlador siempre que no haya una entrada coincidente en la tabla de flujo de un interruptor y el controlador necesita tomar una decisión); y ¿Cuantos mensajes PACKET_IN puede un controlador manejar por segundo? Para ello se evaluó y comparo el rendimiento de controladores OpenSource utilizando una herramienta de benchmarking de código abierto llamada Cbench en sus dos modos de operación: modo de latencia y rendimiento.

Chahlaoui & Dahmouni, (2018) indica que es necesario programar las actividades de administración de tráfico en los factores de retardo y latencia en los controladores. Los autores solucionaron la problemática con flujos QoS de grano fino que permitían automatizar la gestión de múltiples conmutadores en la red OpenFlow.

Laassiri, Moughit & Idboufker, (2018) indica que el controlador es el ente principal de la arquitectura SDN, porque es el encargado de tomar las decisiones sobre el funcionamiento de la red. Basa su investigación en dos reglas: El tiempo necesario para insertar una nueva regla en el interruptor (latencia) y el número de peticiones procesadas por segundo (por ejemplo, rendimiento).

Sanchez & Sarmiento, (2018) resalta quizás una de las deficiencias más importantes encontradas en la arquitectura SDN, se refiere al riesgo de la centralización de la inteligencia del controlador. Se plantea incógnitas de que sucedería si el controlador al ser el único responsable de gestionar e indicar las decisiones de reenvío al plano de datos, presenta fallas en sus decisiones. Como solución se recomienda adoptar topologías de red personalizadas insertando un control total sobre los datos y las interconexiones donde podemos inyectar varios tipos de fallas y monitorear el tráfico.

Priyadarsini, Bera & Bhampal, (2018) indica que la arquitectura SDN ofrece numerosas ventajas y desafíos, si bien es cierto las SDN representan la solución idónea en un entorno real, es importante evaluar y categorizar los parámetros que afectan el rendimiento del controlador SDN, el cómo las variaciones de esos parámetros influyen en su rendimiento.

Pereira, Silva & Sousa, (2019) explica que la transición al paradigma SDN va a resultar en un difícil proceso en la toma de decisiones de los profesionales de red e investigadores. La amplia

diversidad de controladores SDN existentes, así como sus características heterogéneas y potencialidades, son elementos que pueden presentar dudas en la elección del más óptimo en base a un determinado requerimiento. Para ello se realiza un estudio comparativo entre algunos de los controladores.

Mamadou, Azzouni & Pujolle, (2019) nos indica que, si bien es cierto, han aumentado en los últimos meses las contribuciones de investigación, la gran mayoría se centra solamente en que tan veloz se procesa las solicitudes de los hosts.

1.3. Teorías relacionadas al tema.

1.3.1. Desafíos y Tendencias en TIC

Cuando hablamos de desafíos en la red de datos, explícitamente nos podemos referir a su performance o rendimiento que debe ofrecer hacía los requerimientos de los usuarios, ejemplo (velocidad de transmisión, capacidad para trasmitir información de gran tamaño, seguridad, respuesta ante incidentes, entre otros). Hace décadas los grandes avances en tecnologías no habían sido tan influyentes y requeridos como en la actualidad.

Actualmente las redes no solo deben ser robustas en infraestructura, sino también escalables y centralizadas en su gestión de los recursos y dispositivos de red.

Según (Cisco, 2019) la exigencia de ancho de banda en la Internet presenta exponencial consumo y se proyecta que en el presente año 2021 se haya alcanzado los 278 exabytes por mes, lo que implica una nueva evaluación sobre el enfoque de las redes y sus arquitecturas.

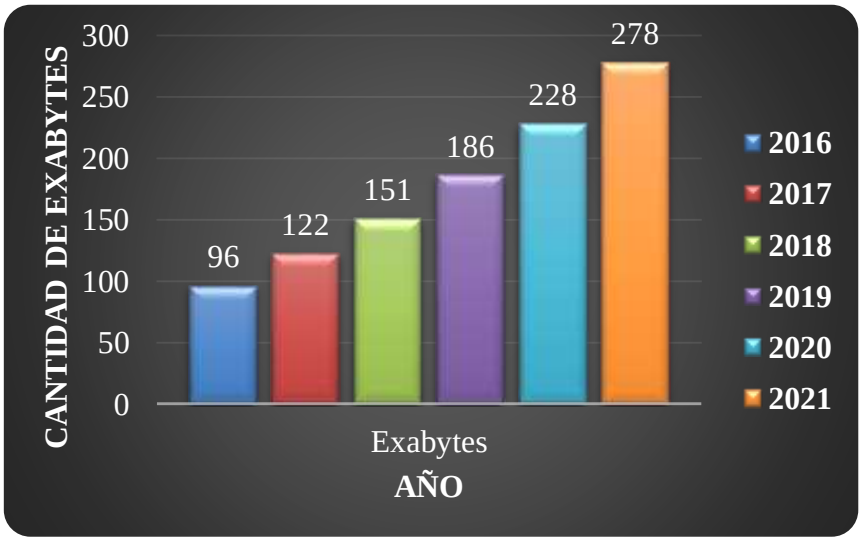

Figura 4. Tráfico de Internet en el año 2021

Con las cifras detalladas anteriormente queda en evidencia que las arquitecturas tradicionales no garantizan el satisfacer los robustos requerimientos en los recursos y tecnologías actuales, este panorama preocupante lo explica Gigaom Resarch:

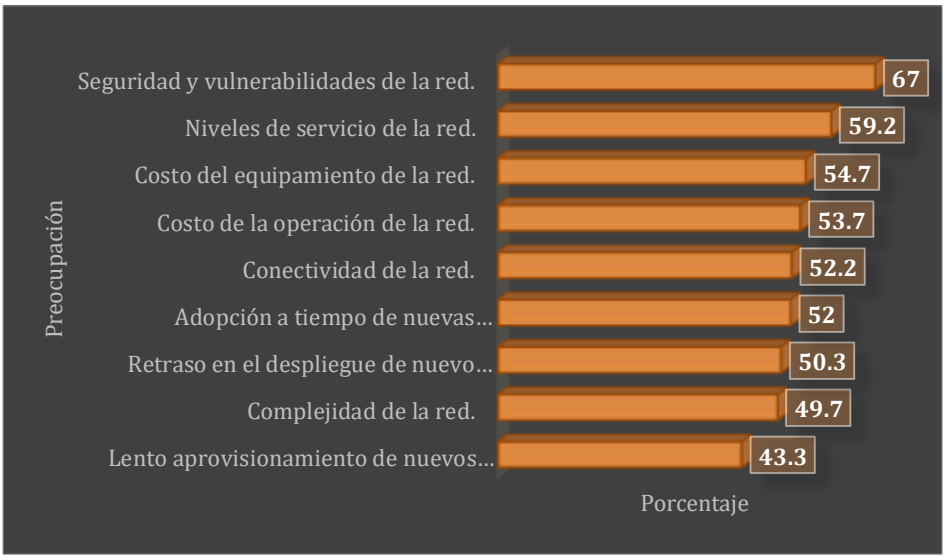

Figura 5. Situaciones preocupantes para las Empresas y Usuarios

1.3.2. Redes basadas en intención

Uno de los ambiciosos e interesantes proyectos de tecnologías orientado a la red de datos, es la que se ofrece para el año 2025, también referida como las “Redes basadas en la intención”, su metodología utiliza el lenguaje natural que permite recibir los requerimientos y de forma automática traducirlos en acciones automatizadas basadas en políticas de calidad de servicio, entre otros. (Cisco, 2019)

1.3.3. Computación en la Nube

En la actualidad sus servicios tienen gran acogida, entre los cuales figuran: Software como servicio (SAAS), se refiere a la adquisición de un Software completo que el proveedor administra y otorga respaldo a los usuarios finales. Plataforma como servicio (PAAS), se encarga de forma recurrente de los Sistemas Operativos o hardware donde se implantan las aplicaciones y su administración, se encargan del mantenimiento, actualizaciones y demás tareas que garanticen la disponibilidad y funcionalidad de los servicios. Infraestructura como servicio (IAAS), ofrece acceso a los componentes de la red y sus funciones, de forma virtual o dedicada para los equipos y la capacidad de almacenamiento de datos, es lo más parecido a un Departamento TI de una empresa. Como ventajas se puede indicar las siguientes: El costo en gastos se reduce de manera considerable, surge la adaptabilidad en base al tipo y crecimiento de la empresa.

Se logra implementar mediante el modelo de la Nube, solución Híbrida e Implementación local en las instalaciones del cliente.

Figura 6. Servicios CLOUD

1.3.4. BIG DATA

El acelerado y constante crecimiento de información en las empresas genera un desafío en el tratamiento de la misma, la velocidad y variedad ha logrado superar las capacidades de almacenamiento con las que se respaldas las empresas. En base a esa necesidad las empresas recurren al análisis de datos en repositorios de datos de gran magnitud y teniendo en cuenta los pilares de integridad, confidencialidad y disponibilidad de los datos.

1.3.5. 5G

Nos referimos a la versión quinta de redes de comunicaciones. Se caracteriza por la velocidad que ofrece, una transferencia de 5000 megabits por segundo aproximadamente. Su objetivo es convertirse en respaldo de la tecnología 4G, que en pocos años estará desfasada.

1.3.6. IoT

La fantasía de lograr que nuestros objetos con los que cotidianamente interactuamos adopten inteligencia y obedezcan nuestras instrucciones es ahora una realidad. La evolución de la Internet ha logrado la conexión y comunicación tecnológica con los mismos, facilitando nuestras actividades diarias y hasta críticas. A su vez se debe seguir trabajando para optimizar el pesado tráfico que generan y ancho de banda que son utilizados por sus protocolos.

1.3.7. BYOD

Desempeñar actividades del trabajo en nuestros dispositivos móviles, se presenta como una tendencia útil, que brinda ayuda en circunstancias donde se requiere acceder a los recursos de las empresas, cerrando la brecha de comunicación y disponibilidad, pero teniendo en cuenta la seguridad y confidencialidad de la información.

1.3.8. Redes Definidas por Software

Cuando hablamos de lenguaje de programación, sintaxis, librerías, etc, relacionamos en gran parte a los sistemas de información. Pero en la era digital emerge un paradigma que cambia la forma de administrar y gestionar una red de computadoras, aquellas que se rigen por topologías estándar en los dispositivos de red, que en la actualidad no satisfacen de forma aceptable los requerimientos.

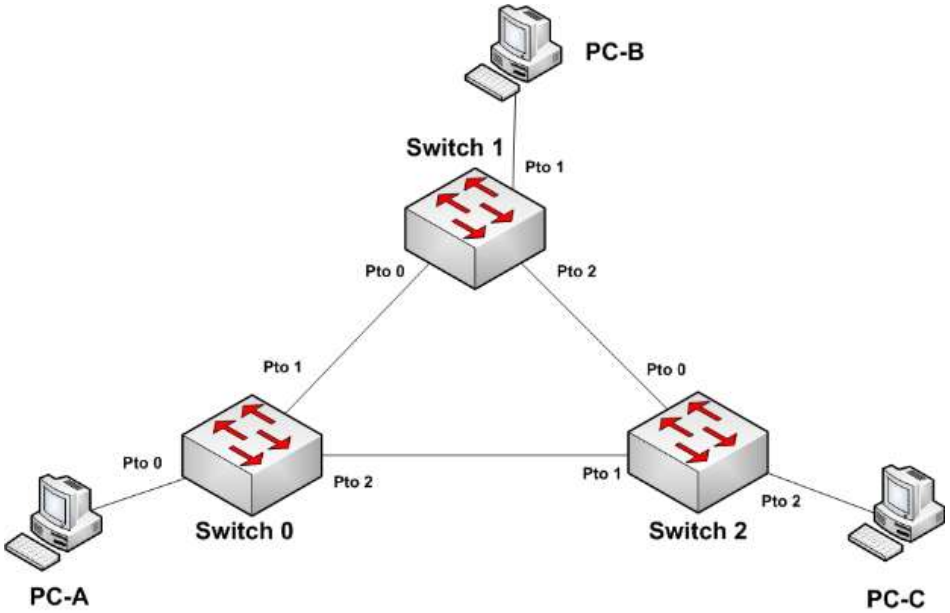

Figura 7. Topología en Paradigma Convencional

El paradigma de las SDN, desencapsula el plano de control y datos respectivamente, abstrayendo las funciones de administración e inteligencia de los dispositivos conmutadores y que mediante su controlador logra centralizar lógicamente la gestión y otorgar una red programable que denote eficiencia, escalabilidad, flexibilidad, entre otras características de la arquitectura.

1.3.8.1. Arquitectura

Según la IETF - RFC 7426. la arquitectura SDN está compuesto por: Plano de Aplicación (Application Plane), Plano de Control (Control

Plane), User/Data Plane, Management Plane, Operation Plane: Interfaces SUR (Southbound Interface), Interfaces al Norte (Northbound Interface).

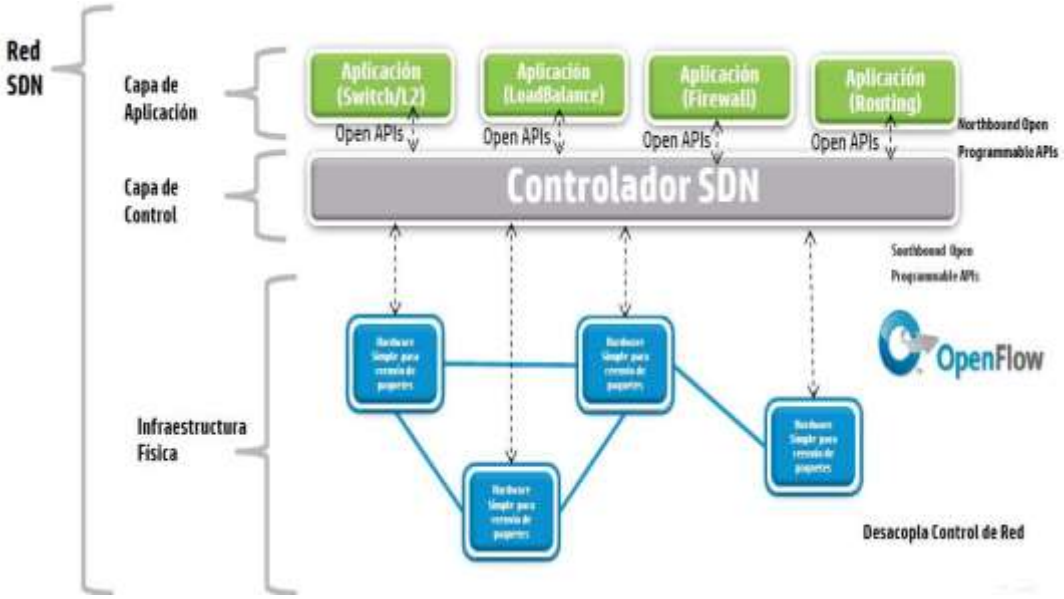

Figura 8. Arquitectura en capas de SDN RFC 7426 (2015)

Tabla 1. *Comparación de las alternativas de implementación*

Modelo SDN / Característica	SDN Open Source (OpenFlow)	SDN via Overlay	SDN via APPIs
Caso de Uso principal	Varios	Centro de datos	Core/WAN
Hardware	Switches OpenFlow	Soporte VXLAN VTEP en servidores o switches	Routers tradicionsles
Protocolo principal	OpenFlow	VXLAN	Segment Routing
Adopción en la industria	Baja	Media	Baja
Siguiente paso habitual en redes tradicionales	Implementar caso de uso específico	Implementar OpenStack, VMWare, NSX, Nuage VSP, Juniper Contral, entre otros.	Conseguir soporte de Segment Routing, PCC y BGP-LS en la red

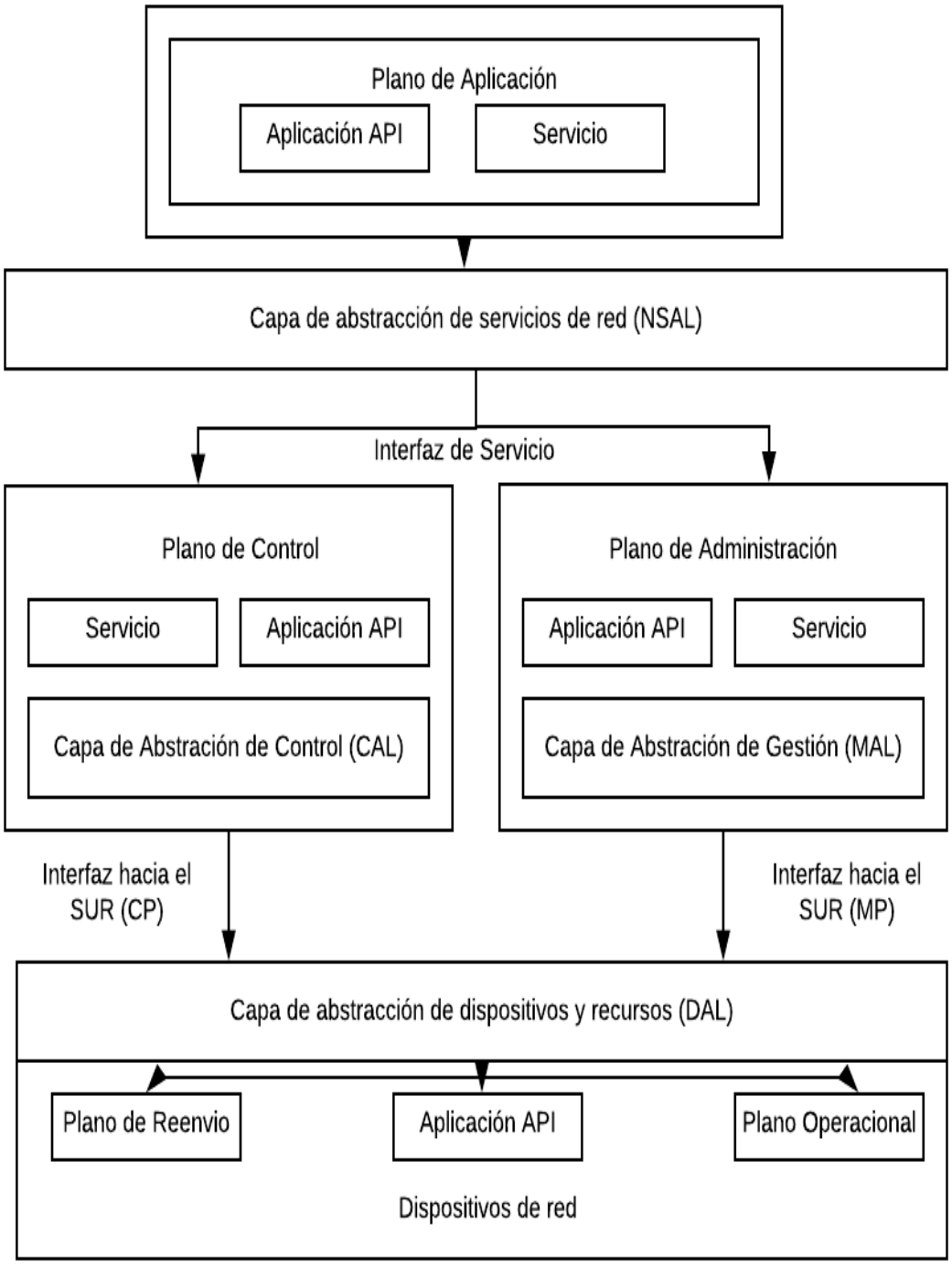

Figura 9. Arquitectura en capas de SDN RFC 7426 (2015)

1.3.9. OpenFlow

Openflow es por ahora el protocolo más estable para la comunicación entre las interfaces y APPs. Se conforma por lo siguiente: Tabla de flujos, Canal seguro, Protocolo OpenFlow. Las acciones que puede realizar el OFS son: 1) Forwarding: Reenvío de flujo de paquetes por un(os) puerto(s) específicos.2) Encrypting: Encapsular y cifrar flujos de paquetes de datos para un controlador. 3) Drop: Borrado de paquetes por seguridad para frenar ataques de DDoS. Estas acciones son consignadas en la tabla de flujos, que tiene 3 campos especiales, donde queda almacenada la información: 1) Header: Encabezado del paquete, define el flujo. 2) Action: La acción. Cómo se procesará el flujo. 3) Statistics: Estadísticas de procesamiento, número de paquetes y por cada flujo.

El funcionamiento es práctico, empieza cuando recibe un paquete que proviene de algún puerto de nuestro Switch, primero realiza una comparación entre la cabecera de dicho paquete con el campo *Match* que se encuentra en su tabla de flujos, previamente configurada para recibir los datos, luego ejecuta las instrucciones que le indica el controlador, entre ellas podemos mencionar, ejecutar Flow Entry o flujo de entrada, verificar el número de prioridad que le indica la cabecera en el campo Priority (de 16 bits), sino encuentra coincidencia con algún dato de su base de datos o conocida tabla de flujos, retorna el paquete al controlador usando un canal seguro, y queda en espera de recibir actualización en sus instrucciones, o en su defecto elimina el paquete.

1.3.10. Mininet

Mininet es una de las herramientas de emulación más usadas para crear topologías de red con los componentes y dispositivos de red habituales: hosts, conmutadores (switch, router), controladores SDN y enlaces. Los hosts de Mininet se basan en Linux, y sus conmutadores

admiten el protocolo OpenFlow para un enrutamiento personalizado. Incluye una aplicación CLI gráfica llamada MINIEDIT, que permite seleccionar y configurar los parámetros de los dispositivos. Por otro lado, ofrece topologías custom, para explorar las diferentes topologías existentes. Pero la característica más resaltante es que permite importar topologías personalizadas realizadas en Python para ejecutar de forma local o controlador remoto.

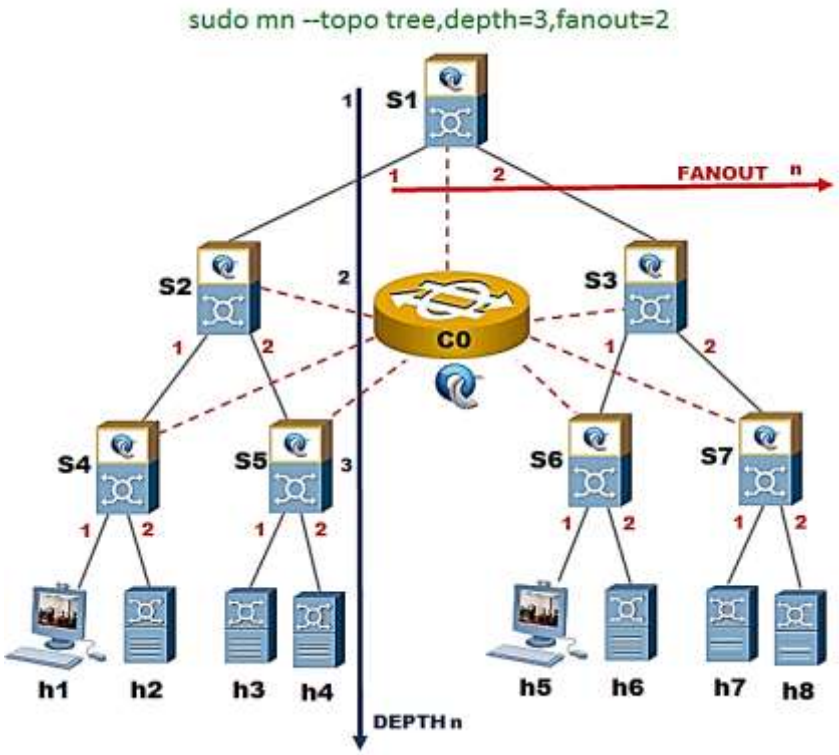

Figura 10. Topología Árbol en Mininet

Las redes MININET hace uso de la virtualización basada en procesos para ejecutar por ejemplo hasta 4096 hosts y conmutadores en un solo núcleo de sistema operativo. Syntaxys: *mininet>[nodo] comando.* Topología propuesta para el estudio: Para generar el banco de pruebas se creó una topología semejante a las actuales de centro de datos. Archivo en Python con su topología.

$sudo mn --custom topología_red.py –topo RedUSS—mac –controller=remote, ip=192.168.1.49, port=6633 – switch=ovsk, protocols=OpenFlow13

1.3.11. Controlador SDN

En el paradigma de las Redes Definidas por Software (SDN), el controlador es el componente que dentro de sus funciones incluye la gestión, orquestación, administración o centralización de la inteligencia de la red, se fracciona en los componentes: Infraestructura, Control y Aplicaciones. Su principal objetivo se centra en optimizar el flujo o rendimiento de la red en su totalidad, desde los dispositivos de red hasta las Apps que en su conjunto forman una potente herramienta tecnológica.

El controlador SDN inicia el flujo comunicándose con los switches Openflow para indicar el camino de enrutamiento del tráfico de la red. Utiliza las Interfaces hacia el SUR, aquella que se encarga de gestionar las misivas entre los switches OpenFlow y el controlador SDN. Asimismo, las interfaces hacia el NORTE se encargan de notificar el controlador con aplicaciones externas, también llamadas APIs. En el intermedio se encuentra la capa de abstracción de servicio (SAL).

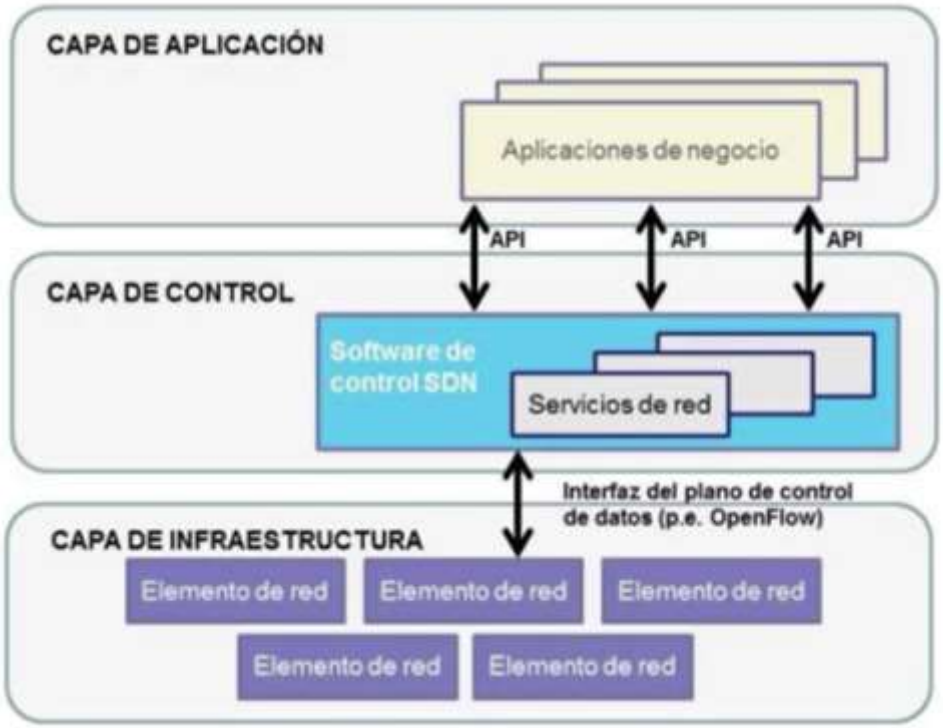

Figura 11. Componentes en la Arquitectura de un controlador SDN.

En la actualidad la programabilidad llego a las redes de datos, y con ella la fascinante oportunidad de adquirir experiencia en la implementación de controladores. En el mercado se ofrecen diversos controladores de código abierto y propietario. Pero aquello también representa una dificultad, debido a los constantes lanzamientos presenta un desafío el poder seleccionar el más adecuado y adaptable a la realidad de la empresa, puesto que la mayoría no aporta sólida y detallada documentación en cuanto a despliegues y casos de éxito en entornos reales.

Tabla 2. *Principales controladores SDN de código abierto.*

Controlador	**Fuente**	**Licencia**	**Lenguaje**
OpenDayLight	OpenDayLight	EPL	Java
ONOS	ON.Lab	Apache 2.0	Java
NOX	ICSI	GPL	C++
POX	ICSI	GPL	Python
Beacon	Stanford University	GPLv2, Stanford FOSS	Java
Floodlight	Big Switch Networks	Apache 2.0	Java
Ryu	NTT Communications	Apache 2.0	Python
FLowER	Travelping GmbH	MIT License	Erlang
Jaxon	University of Tsukuba	GPLv3	Java
Mul SDN	Kulcloud	GPLv2	C
NodeFlow	Gary Berger	-	Javascript
Trema	NEC	GPLv2	Ruby.C

1.3.12. OpenDayLight

OpenDaylight (ODL) Es por ahora uno de los proyectos más estables y ventajosos para utilizar en un proyecto SDN. Se soporta en un lenguaje de Código Abierto, lo que facilita e incrementa su uso en implementaciones de proyectos SDN, utiliza protocolos OpenSource para monitorear, programar y centralizar la gestión de los dispositivos de red de la capa de datos, resalta su compatibilidad con el protocolo OpenFlow, así como features o soluciones para instalar desde su repositorio que son de gran utilidad, por ejemplo, para el inventario de equipos. Asimismo, presenta una interfaz amigable que permite el monitoreo en tiempo real de la topología y actualiza la inserción o eliminación de nodos o conmutadores.

Entre las versiones más notables en despliegue figuran Hydrogeno (FEB2014), Helium (OCT2014), Lithium (JUN2015), Berylium (FEB2016), Boron (NOV2016), Carbon (JUN2017), NITROGEN (SEP2017), OXYGEN (MAR2018), FLUORINE (AGO2018), NEON (MAR2019), entre otros.

OpenDaylight Sodium es la undécima versión de la plataforma controladora SDN de código abierto más segura, estable, escalable y eficaz. Contiene una nueva funcionalidad para reforzar el liderazgo de OpenDaylight en casos de uso clave que incluyen conectividad / enrutamiento WAN y redes virtuales en entornos de nube y perimetrales, junto con mejoras de estabilidad y escalabilidad.

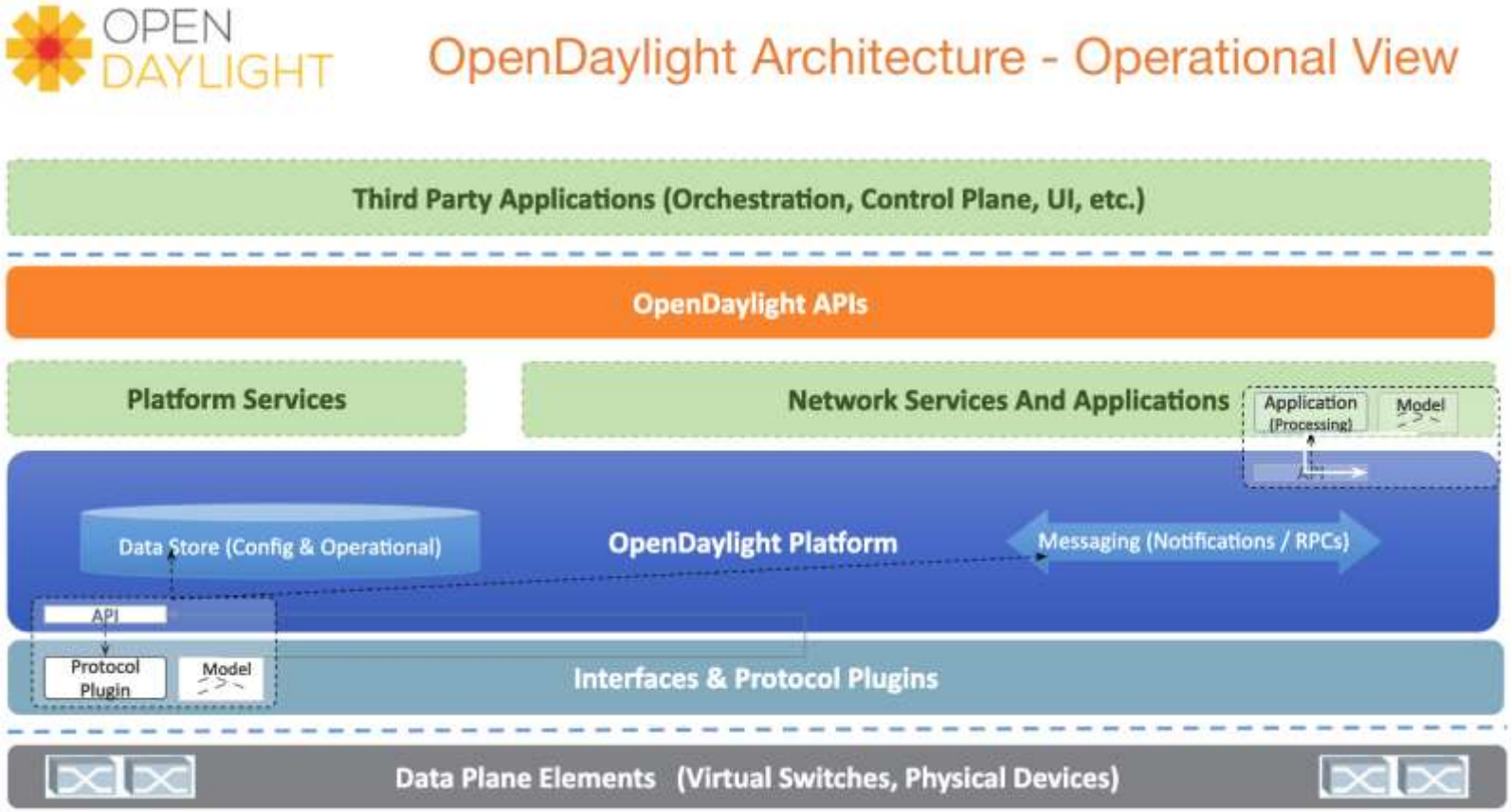

Figura 12. Arquitectura del controlador Opendaylight

1.3.13. ONOS

ONOS son las siglas de Open Network Operating System. ONOS otorga el plano de control para una arquitectura SDN, entre sus bondades podemos indicar la administración de los componentes de red, entre conmutadores y enlaces, y ejecuta programas o módulos de software para proporcionar servicios de comunicación a los hosts finales y redes distintas.

El kernel de ONOS, servicios centrales y aplicaciones están escritos en Java como paquetes que se cargan en el contenedor Karaf OSGi. OSGi es un sistema de componentes para Java que permite que los módulos se instalen y ejecuten dinámicamente en una sola JVM.

La Línea primoridal en la arquitectura ONOS son sus servicios, que a su vez está conformado por subsistemas que pertenecen a los niveles: Aplicación (App), Gestor (Manager) o Proveedor (Provider).

Entre sus diversas versiones destacan las más actualizadas, tales como Velociraptor, Uguisu, Toucan, Sparrow, Raven, Quail, etc.

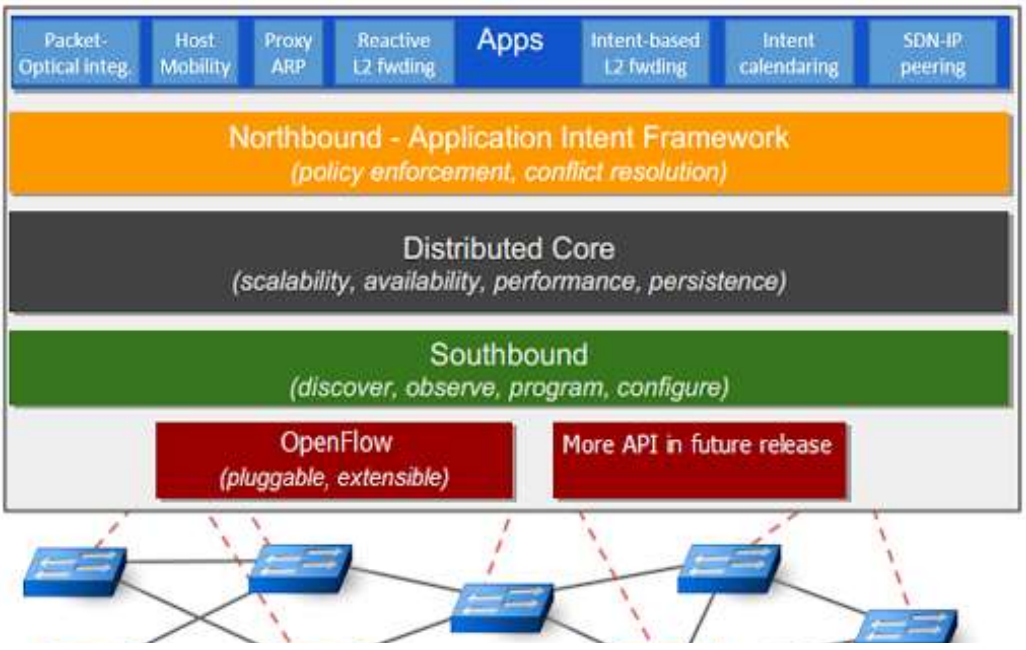

Figura 13. Arquitectura del controlador ONOS

1.3.14. GNS3

GNS3 es un software que permite emular redes personalizadas, con sus respectivas configuraciones, testeos de conectividad, consolida el proyecto de Redes de Datos, para probar casuísticas de incidencias en las redes virtuales y reales. útil para la creación y despliegue de los controladores SDN. GNS3 consta de dos componentes de software: Software GNS3 todo en uno (GUI) y Servidor/Máquina Virtual GNS3.

1.3.15. CBENCH

Cbench (Controller Benchmarker) forma parte de las herramientas de testing con mayor acogida por los investigadores y profesionales que pretenden realizar análisis en el ámbito de redes de datos. Su principal función es emular de forma personalizada la cantidad de conmutadores OpenFlow que se comunican con un controlador OpenFlow con el objetivo para medir diferentes aspectos del rendimiento y latencia del mismo. La prueba envía mensajes de aviso de llegada de nuevos flujos (mensajes OFPT_PACKET_IN) teniendo como origen los conmutadores hacia el controlador, aquel que monitorea la respuesta. (OFPT_PACKET_OUT o OFPT_FLOW_MOD).

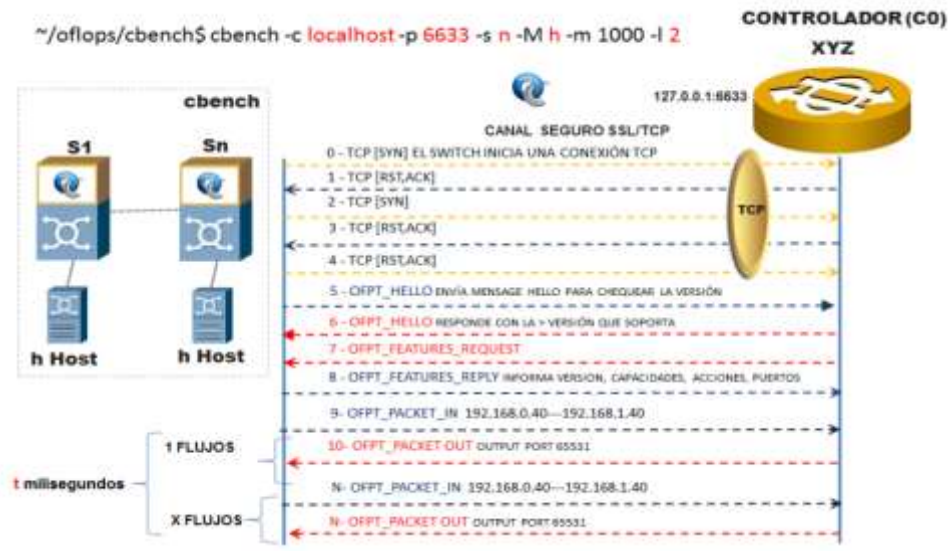

Figura 14. Funcionamiento de la Herramienta CBENCH

1.4. Formulación del Problema.

¿Qué controlador de Redes Definidas por software de tipo Open Source, es el que tiene mejor rendimiento en la gestión de una red de computadoras?

1.5. Justificación e importancia del estudio.

El tema de esta investigación radica en el alcance y ventajas que ostentan las redes programables, SDN ofrece la optimización mediante la centralización de la gestión de una red de computadoras utilizando su núcleo o controlador, el cual requiere una evaluación comparativa basado en métricas o factores de rendimiento. Por otro lado, el auge del paradigma genera múltiples versiones, dejando como déficit la falta de documentación básica para la implementación y banco de pruebas.

En este trabajo se evaluará y comparará los funcionamientos de los controladores SDN de código abierto más prominente (ONOS y OpenDayLight) considerando la latencia y el rendimiento como clave métrica, motivado por los beneficios adicionales que representa, como lo es la alta disponibilidad, ya que para la gran mayoría de los servicios de una empresa tener un sistema sin alta disponibilidad es algo obsoleto; siendo así, evaluar las prestaciones y funcionamiento resulta de gran importancia.

1.6. Hipótesis.

H0: El controlador de Redes definidas por Software de tipo Open Source ONOS ofrece un mayor rendimiento y una menor latencia que el controlador OpenDaylight en la gestión de una red de computadoras.

1.7. Objetivos

1.7.1. Objetivos General

Comparar el rendimiento de controladores de Redes definidas por Software de tipo Open Source para la gestión de una red de computadoras mediante el control centralizado.

1.7.2. Objetivos Específicos

1. Seleccionar controladores SDN Open Source.
2. Diseñar un entorno de pruebas.
3. Desplegar el entorno en base al diseño seleccionado.
4. Implementar los controladores antes seleccionados.
5. Realizar experimentos de rendimiento, con el uso de herramientas Open Source.
6. Análisis de resultados.

II. MATERIAL Y MÉTODO

2.1. Tipo y Diseño de Investigación.

La característica del actual trabajo es Cuantitativa, ya que se fundamenta en el aplicación y estudio de la realidad a través de diferentes operaciones establecidas en la medición. Se soportará en experimentos y conseguirá hallazgos diferidos a partir de la hipótesis. Asimismo, es de tipo comparativa aplicada, dado a que en esta investigación se realizará la comparación de controladores de redes definidas por software, con el fin de evaluar que controlador SDN es el que tiene mejor rendimiento en la gestión de una red de computadoras. El diseño de esta investigación es de tipo Experimental y Sub tipo Experimento Puro.

2.2. Población y muestra.

2.2.1. Población

La población de este tema de investigación son los diferentes controladores de redes definidas por software existentes para gestión de redes de computadoras.

Tabla 3. *Comparación de Controladores según criterios de evaluación*

CARACTERÍSTICA	SUB CARACTERÍSTICA	PUNTAJE ASIGNADO	CONTROLADORES OPEN SOURCE									
			ONOS	ODL	NOX	POX	RYU	BEACON	MAESTRO	FLOODLIGHT	IRIS	MUL
LENGUAJE DE PROGRAMACIÓN	JAVA	3	3	3	-	-	-	3	3	3	3	-
	PYTHON	2	-	-	-	2	2	-	-	-	-	-
	C++	1	-	-	1	-	-	-	-	-	-	1
GUI (Interfaz Gráfica)	Web Based	3	3	3	-	-	-	3	-	3	3	3
	Python + QT4	2	-	-	2	2	2	-	-	-	-	-
DOCUMENTACIÓN	Básica	1	-	-	-	-	-	-	1	-	-	-
	Buena	3	3	3	-	-	-	-	-	-	-	-
	Regular	2	-	-	-	-	2	2	-	2	-	-
	Pobre	1	-	-	1	1	-	-	1	-	1	1
MODULARIDAD	Alta	3	3	3	-	-	-	-	-	-	-	-
	Regular	2	-	-	-	-	2	2	2	2	2	2
	Baja	1	-	-	1	1	-	-	-	-	-	-
CENTRALIZADA / DISTRIBUIDA	Distribuida	2	2	2	-	-	-	-	-	-	-	-
	Centralizada	1	-	-	1	1	1	1	1	1	1	1
PLATAFORMA DE SOPORTE	Linux	3	3	3	3	-	3	3	3	3	3	3
	Windows	2	2	2	-	2	-	2	2	2	2	-
	MAC OS	1	1	1	-	1	-	-	1	1	1	-
SOPORTE OPENFLOW	OF 1.3 o 1.4	3	3	3	-	-	3	-	-	3	3	3
	OF 1.2	2	-	-	-	-	2	-	-	-	-	2
	OF 1.0, 1.1	1	1	1	1	1	1	1	1	1	1	1
INTERFAZ SB	SI	1	1	1	-	-	1	-	-	-	1	1
	NO	0	-	-	-	-	-	-	-	-	-	-
INTERFAZ NB	SI	1	1	1	1	1	1	1	1	1	1	1
	NO	0	-	-	-	-	-	-	-	-	-	-
REST API	SI	1	1	1	1	1	1	1	1	1	1	1
	NO	0	-	-	-	-	-	-	-	-	-	-
	SI	1	1	1	1	1	1	1	1	1	1	1

PARTNER RECONOCIDOS	NO	0	-	-	-	-	-	-	-	-	-	-
SOPORTE DE MULTIHILO	SI	1	1	1	-	1	1	1	1	1	1	1
	NO	0	-	-	-	-	-	-	-	-	-	-
SOPORTE OPENSTACK	SI	1	-	1	-	-	1	-	-	-	-	1
	NO	0	-	-	-	-	-	-	-	-	-	-
DOMINIO DE LA APLICACIÓN	WAN-DataCenter	3	3	3	-	-	-	-	-	-	3	3
	Campus	2	2	-	2	2	2	-	-	2	-	-
	Transporte	1	1	-	-	-	-	-	-	-	-	-
TOTALES			**35**	**33**	**15**	**17**	**26**	**21**	**19**	**27**	**28**	**26**

2.2.2. Muestra

La muestra está especificada por dos de una población de diez controladores de Redes Definidas por Software de tipo Open Source (mayor puntuación): ONOS y OPENDAYLIGHT.

2.3. Variables, Operacionalización.

2.3.1. Variables

2.3.2. Variable Dependiente

Gestión de Redes de Computadoras

2.3.3. Variable Independiente

Controlador SDN

2.3.4. Operalización

Variable independiente	Dimensiones	Indicadores	Unidad de Medida	Técnicas e instrumentos de recolección de datos
Controlador SDN de tipo Open Source: x: ONOS y: OpenDayLigth	Tiempo	Conectividad entre nodos	Mbits/sec - ms	Análisis documental. Experimentos de Laboratorio Pruebas de Rendimiento.
		Verificación del intercambio de mensajes OpenFlow	Paquetes	
		Pérdidas de paquetes	Porcentaje %	
		Latencia y variación de la latencia - Throughput	ms	
	Capacidad	Cantidad de Menoría disponible	Unidad	
		Consumo de CPU	Porcentaje %	
		Consumo de memoria RAM	Porcentaje %	
Variable dependiente	**Dimensiones**	**Indicadores**	**Unidad de Medida**	**Técnicas e instrumentos de recolección de datos**
Gestión de Redes de Computadoras	Disponibilidad	El porcentaje de tiempo que el controlador está en marcha y disponible.	Número	

	Rendimiento	Razón de establecimiento de flujos (i)=(Cantidad de respuestas OF (i))/(Duración de la prueba(i))		Análisis documental. Experimentos de Laboratorio. Pruebas de Rendimiento.
	Escalabilidad	Cantidad de switches y hosts que el controlador puede manejar $: cbench -c <IP del controlador> -p<puerto > -s <cantidad de switch> -M <número de host> -m <duración de la prueba en milisegundos> -l <cantidad de pruebas > -t	Número	

2.4. Procedimientos de análisis de datos.

2.4.1. Método de Observación Directa

Los observadores pueden recolectar datos a través de notas de campo, grabaciones de vídeo o de audio, que pueden analizarse mediante herramientas de análisis cualitativo. Si se codifican las observaciones para obtener datos numéricos exactos, pueden analizarse mediante un enfoque cuantitativo.

La observación directa permite acceder y observar cómo se gestiona las redes de datos en la actualidad, la problemática recurrente y los momentos de congestionamiento y tráfico alto, que disminuye la calidad de servicio.

2.4.2. Método de Análisis bibliográfico

Libros y la Internet. – Son aquellos que facilitan los datos necesarios para el proceso de investigación. Documentos de la USS.

2.4.3. Método de Encuesta y Entrevista

Las entrevistas son una herramienta principalmente para la recolección de datos cualitativos y son populares como instrumentos para recopilar información debido a su flexibilidad.

Las interacciones que se generan en una entrevista pueden presentarse de forma estructurada o semiestructurada para generar ideas y conceptos, se tienen en cuenta los siguientes factores: Exhaustividad, Tacto, Precisión, Exactitud y Confidencialidad.

Las entrevistas se realizaron a profesionales responsables de la Administración / Gestión de las áreas de Redes y Telecomunicaciones de pequeñas y medianas empresas de la Región.

Como fuente de información se requiere que la Universidad Señor de Sipán, facilite los planos, topología de la red, número de equipos con los que cuenta la universidad, ancho de banda asignado a cada Facultad y departamentos, número de empleados, docentes y alumnos que están actualmente.

2.4.4. Experimentos de Laboratorio

La característica del actual trabajo es Cuantitativa, ya que se fundamenta en el aplicación y estudio de la realidad a través de diferentes operaciones establecidas en la medición. Se soportará en experimentos y conseguirá hallazgos diferidos a partir de la hipótesis. Asimismo, es de tipo comparativa aplicada, dado a que en esta

investigación se realizará la comparación de controladores de redes definidas por software, con el fin de valuar que controlador SDN es el que tiene mejor productividad en la gestión de una red de computadoras. El diseño de esta investigación es de tipo Experimental y Sub tipo Experimento Puro.

2.4.5. Pruebas de Rendimiento.

Se otorgará un análisis detallado de la prueba de rendimiento. Primero se diseñará y ejecutará la configuración de rendimiento y entonces los resultados. Por último se discute resultados obtenidos..

2.4.6. Instrumento de recolección de datos

Documentos de investigación.
Ficha de Observación.
Hojas de Registro.
Entrevistas y Observaciones.
Experimentos de Laboratorio.
Reportes estadísticos de resultados Wireshark, Cbench, Iperf

2.5. Criterios éticos

Consistencia: La investigación se basa en data exacta, el banco de pruebas fue llenado con hojas de cálculo para garantizar la integridad de los resultados.

Validez: El resultado de esta investigación es válido y otorga vigencia de 4 años, debido a que según el consolidado bibliográfico, la mayoría de las investigaciones se realizan con versiones “estables” que datan del año 2014, lo cual se considera estaría desaprovechando los lanzamientos oficiales de controladores con sus respectivas mejoras.

Fiabilidad: Es fiable porque se basa en resultados calculados automáticamente y verificados antes de generar los gráficos y el consolidado.

Transferibilidad: Se busca que la información de esta investigación pueda ser reutilizada y mejorada. generando conocimiento y experiencia en los administradores de red.

2.6. Criterios de Rigor científico.

Coherencia Metodológica: El proyecto cumplirá con lo proyectado en la propuesta.

Validez: Datos obtenidos para la investigación serán evaluados y tratados para poder hacer semejante la realidad en el crecimiento de información usada en una base de datos en la actualidad.

Consistencia: La investigación representará material consistente y certificado por la comunidad científica. Los cuales han sido recopilados y seleccionados para aportar un mejor enfoque a la investigación.

III. RESULTADOS

A. PRUEBAS DE CONFIGURACIÓN:

Las pruebas cumplen un papel fundamental en generar resultados que sirvan para su posterior evaluación y entender el funcionamiento de un proyecto SDN real.

CONECTIVIDAD ENTRE NODOS.

- Ping: Comando que utiliza el envío de paquetes ICMP para testear la satisfactoria conexión de la red, y por ende verificar la

conmutación en los dispositivos de red que utilizan el módulo L2. Switch.

- Ping all: Permite testear la conectividad de forma global, pues emite hacia todos los dispositivos de red ping que permite recabar la información general de la topología.
- Iperf: Es una herramienta multiplataforma que ofrece las funcionalidades de cliente y servidor, permite procesar flujos de datos para calcular el rendimiento entre los dos extremos en una o ambas direcciones. Es ampliamente usado para generar los bancos de pruebas.

VERIFICACIÓN DE MENSAJES OPENFLOW.

Mediante la herramienta Wireshark se verifico el flujo de paquetes OpenFlow.

Para la creación de la red se dispone del controlador ODL con ip ***192.168.122.29*** y Mininet. El comando a ejecutar en Mininet para crear la red es el siguiente:

$ sudo mn –controller=remote,ip=192.168.122.29, --topo=tree,depth=2,fanout=3 –switch=ovsk,protocols=OpenFlow13 –mac

```
mininet> sh ovs-ofctl -O OpenFlow13 dump-flows s1
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x2b0000000000003e, duration=3034.775s, table=0, n_packets=918, n_bytes=
64764, priority=2,in_port=3 actions=output:1,output:2
 cookie=0x2b0000000000003c, duration=3034.775s, table=0, n_packets=918, n_bytes=
64764, priority=2,in_port=1 actions=output:2,output:3
 cookie=0x2b0000000000003d, duration=3034.775s, table=0, n_packets=918, n_bytes=
64764, priority=2,in_port=2 actions=output:1,output:3
 cookie=0x2b0000000000000a, duration=3047.137s, table=0, n_packets=1829, n_bytes
=156685, priority=100,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x2b0000000000000a, duration=3047.137s, table=0, n_packets=0, n_bytes=0,
 priority=0 actions=drop
mininet>
```

Figura 16 - Tabla de flujos del switch1

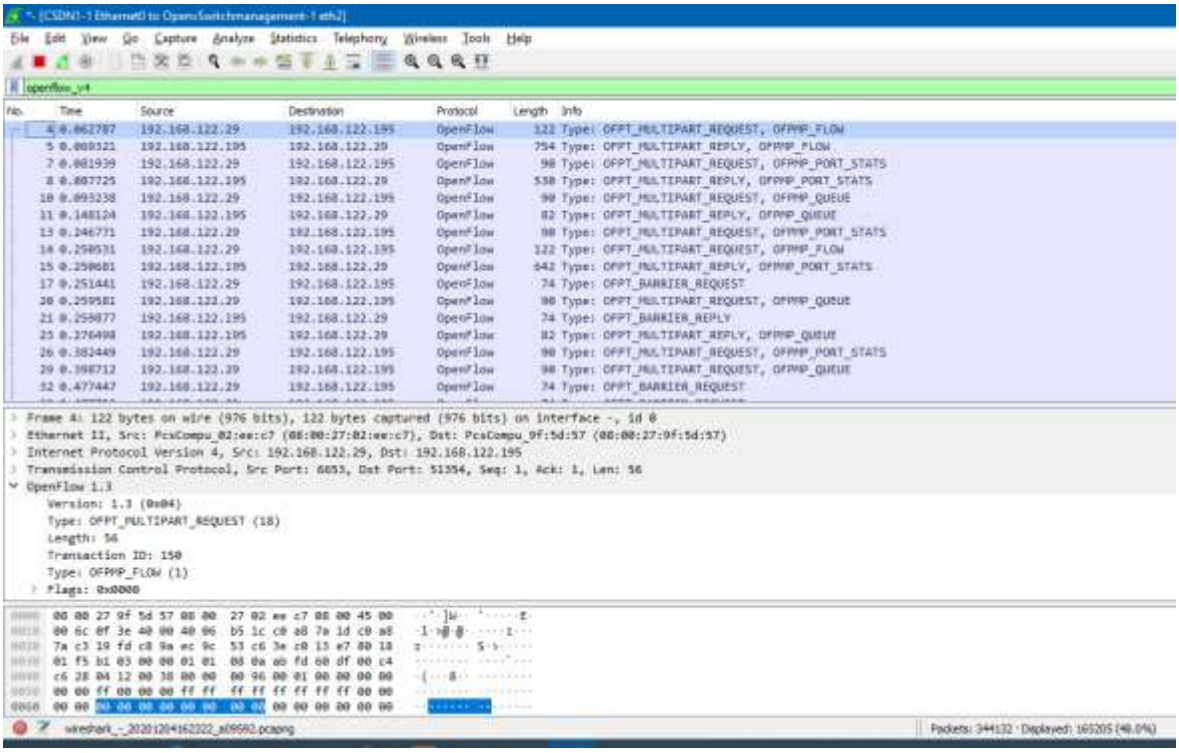

Figura 15 - Captura de paquetes Openflow

Se evidencia el flujo de tráfico de los paquetes enviados desde los hosts hacia los controladores SDN.

B. PRUEBAS DE DESEMPEÑO APLICADA A LAS SDN.

PÉRDIDA DE PAQUETES

Utilizaremos el conocido comando PING, que permite visualizar datos informativos conocidos como mensajes ICMP.

PRUEBAS ICMP Y FLUJOS OPENFLOW

En esta prueba se busca medir los tiempos de respuesta entre los hosts de la red personalizada en MIninet.

Tabla 4. *Direcciones IP y MAC de los hosts simulados.*

NOMBRE DEL HOST (NODO)	IP/MASK	MAC
h1	10.0.0.1/8	00:00:00:00:00:01
h2	10.0.0.2/8	00:00:00:00:00:02
h3	10.0.0.3/8	00:00:00:00:00:03
h4	10.0.0.4/8	00:00:00:00:00:04
h5	10.0.0.5/8	00:00:00:00:00:05
h6	10.0.0.6/8	00:00:00:00:00:06
h7	10.0.0.7/8	00:00:00:00:00:07
h8	10.0.0.8/8	00:00:00:00:00:08
h9	10.0.0.9/8	00:00:00:00:00:09
h10 (Sede Central)	10.0.0.10/8	00:00:00:00:00:0A
h11 (Servidor)	10.0.0.11/8	00:00:00:00:00:0B
h12 (Servidor)	10.0.0.12/8	00:00:00:00:00:0C
h13 (Servidor)	10.0.0.13/8	00:00:00:00:00:0D

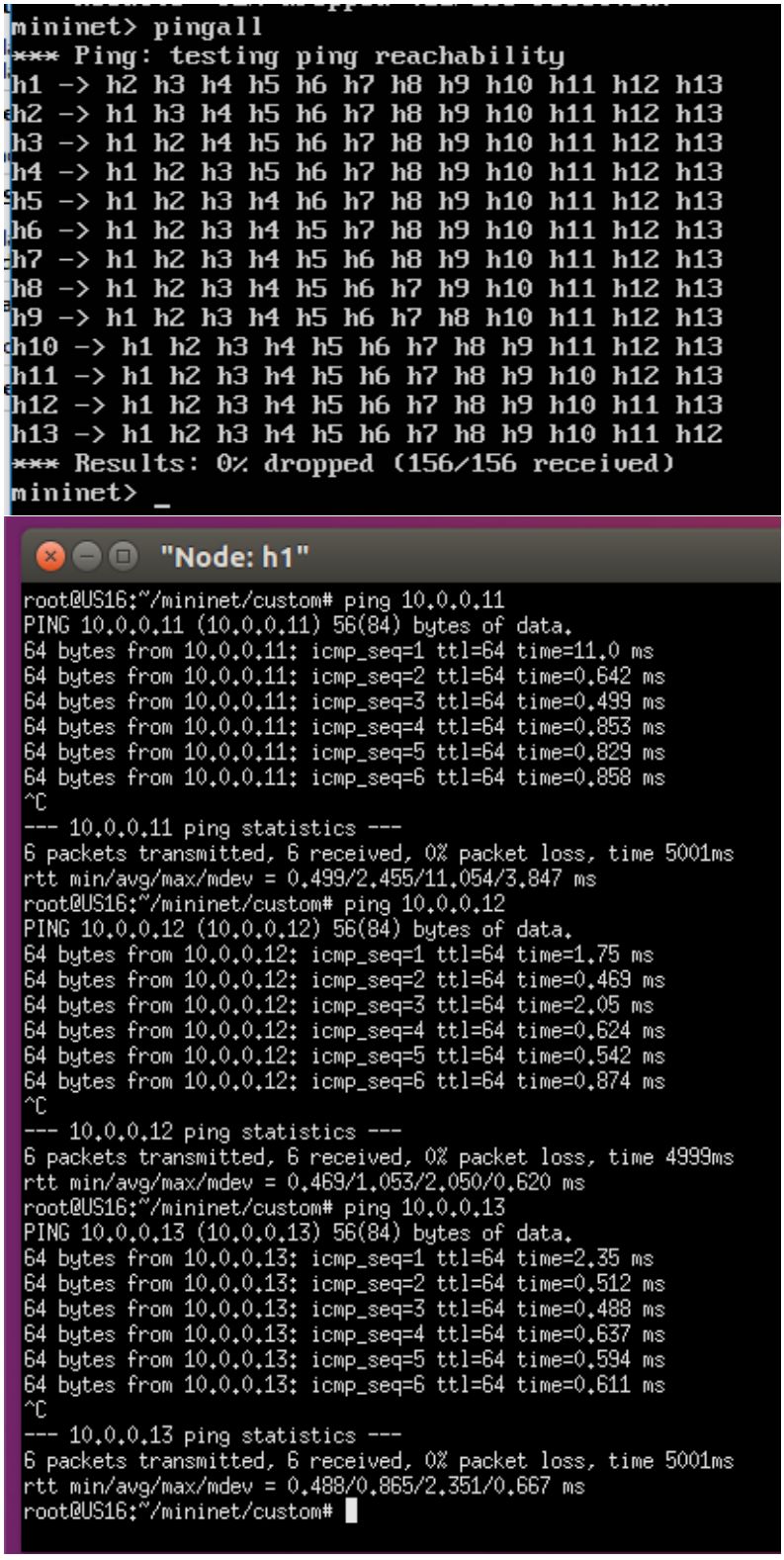

Figura 17. Pruebas de pingall

La comunicación es satisfactoria al 100%, se realizó pruebas recurrentes manteniéndose el resultado satisfactorio.

Tabla 5. *Tiempo de respuesta en milisegundos hacia el host 11*

NOMBRE DEL HOST (NODO)	TIEMPO DE RESPUESTA ICMP HACIA EL HOST H11 (mseg)				
	1ra	2da	3ra	4ta	5ta
H1	11.000	0.642	0.499	0.853	0.829
H2	2.030	0.829	0.414	0.691	0.405
H3	3.290	0.751	0.405	0.372	0.362
H4	3.050	0.731	0.586	0.995	0.703
H5	3.170	0.469	0.327	0.319	0.356
H6	6.930	0.302	0.300	0.314	0.492
H7	9.390	0.450	0.329	0.362	0.408
H8	2.090	0.319	0.536	0.362	0.484
H9	2.040	0.327	0.318	0.436	0.523
H10	1.360	0.447	0.412	0.263	0.353
PROMEDIO	4.435	0.527	0.413	0.497	0.492

Tabla 6. *Tiempo de respuesta en milisegundos hacia el host 12*

NOMBRE DEL HOST (NODO)	TIEMPO DE RESPUESTA ICMP HACIA EL HOST H12 (mseg)				
	1ra	2da	3ra	4ta	5ta
H1	1.750	0.469	2.050	0.624	0.542
H2	8.580	0.445	0.984	0.457	0.568
H3	1.380	1.710	0.491	0.693	0.406
H4	0.811	0.691	0.357	0.872	0.586
H5	1.200	0.491	0.439	0.369	0.310
H6	1.110	0.479	0.301	0.733	0.442
H7	1.250	0.340	0.310	0.401	0.317
H8	1.210	0.306	0.290	0.304	0.388
H9	1.170	0.353	0.369	0.353	0.591
H10	1.220	0.566	0.365	0.325	0.458
PROMEDIO	1.968	0.585	0.596	0.513	0.461

Tabla 7. *Tiempo de respuesta en milisegundos hacia el host 13*

NOMBRE DEL HOST (NODO)	TIEMPO DE RESPUESTA ICMP HACIA EL HOST H13 (mseg)				
	1ra	2da	3ra	4ta	5ta
H1	2.350	0.512	0.488	0.637	0.594
H2	11.100	1.480	0.442	0.679	0.543
H3	0.763	1.230	0.446	1.210	0.327
H4	1.150	0.381	0.407	1.070	0.567
H5	6.060	0.470	0.398	0.520	0.339
H6	1.420	0.429	0.310	0.298	0.481
H7	1.700	0.316	0.290	0.388	0.271
H8	1.560	0.361	0.304	0.310	0.453

H9	1.480	0.493	0.321	0.275	0.390
H10	1.410	0.315	0.333	0.336	0.373
PROMEDIO	2.899	0.599	0.374	0.572	0.434

Según lo descrito en las tablas, el primer paquete nos presenta un alto tiempo de respuesta, con los siguientes la cifra va en disminución, y el motivo radica en que la tabla de flujos del Switch OpenFlow, se encuentra vacía, lo que se traduce en tiempo de retraso. Se indica que el primer paquete enviado es un ARP que se hace llegar al controlador por medio del mensaje "PACKET IN", es una propagación inicial y absoluta. Luego es el controlador Opendaylight que utilizando el módulo L2-SWITCH que fue instalado con el comando *"feature:install"* procesa la solicitud broadcast. Luego el controlador mediante los mensajes "PACKET OUT" replica el mensaje por todos los puertos disponibles, con excepción del puerto que recibió el acuse.

Se pudo comprobar el rendimiento del controlador Opendaylight que instalo correctamente las entradas de flujos en los switch Openflow de forma automática y sin intervención a nivel local del dispositivo de red.

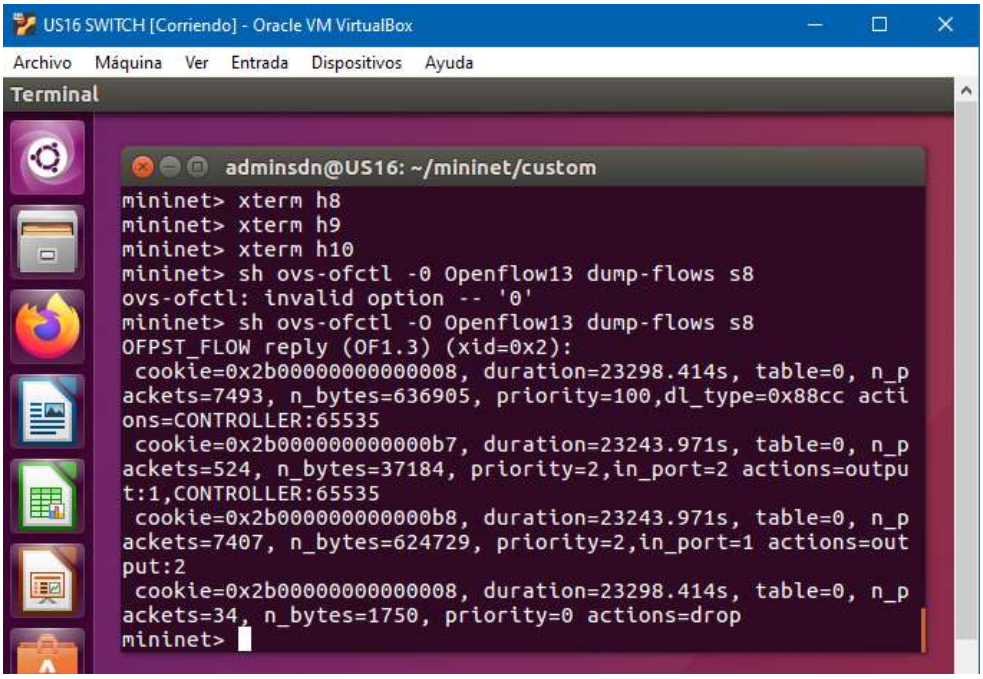

Figura 18. Ingreso de flujos entre el controlador y switch SDN

PRUEBA DE MENSAJERIA ICMP

El objetivo es verificar el funcionamiento de las entradas de flujos necesarias para la comunicación.

Tabla 8. *Asignación de IPs y MACs.*

Nombre del Host (Nodo)	IP	MAC
h1	192.168.1.10/24	00:00:00:00:00:01
h4	10.0.0.4/8	00:00:00:00:00:04
h12 (Red Servidores)	192.168.1.120/24	00:00:00:00:00:0C

El módulo L2-Switch logra instalar entradas o flujos que permiten la misiva entre los hosts y se logró obtener como resultado satisfactorio la conectividad entre los hosts y sin extravió de paquetes.

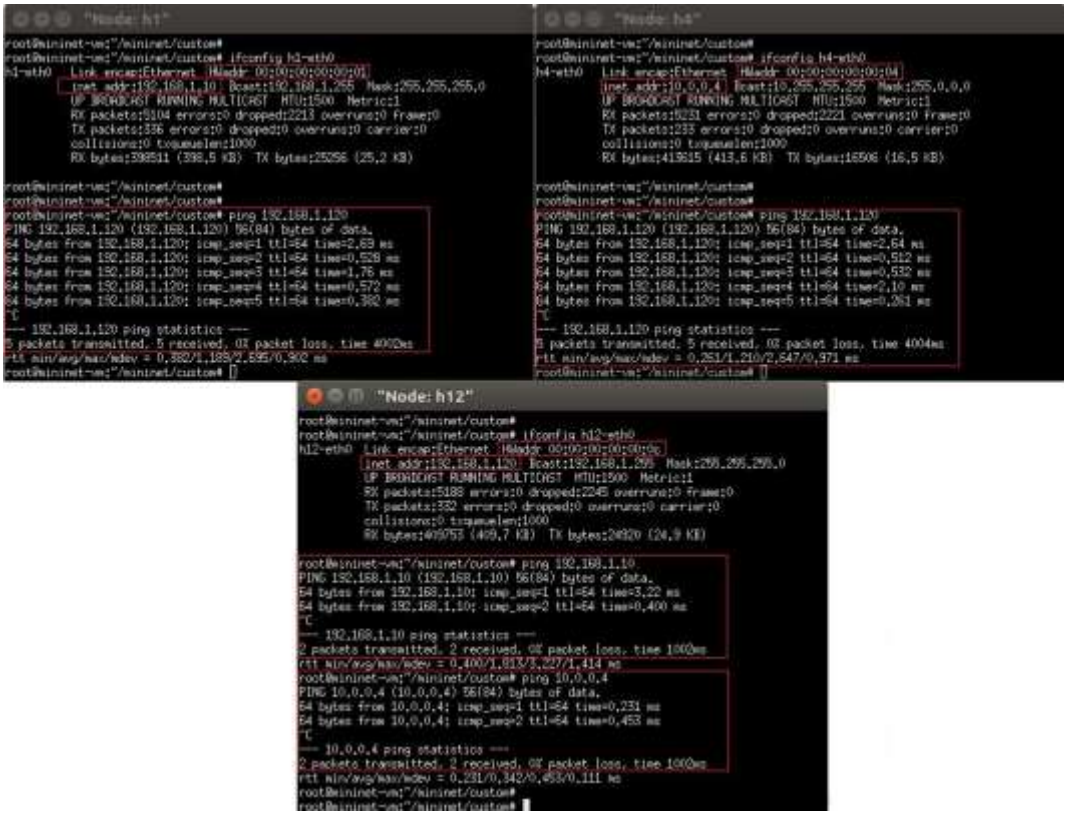

Figura 19. Conectividad entre hosts de diferentes redes.

LATENCIA - VARIACIÓN DE LA LATENCIA – EXTRAVIO DE DATOS

Para la prueba se utilizó el comando PING, que emite valores estadísticos en base a los siguientes valores: mínimo, promedio, máximo y la desviación estándar del parámetro RTT durante el intercambio de mensajes ICMP.

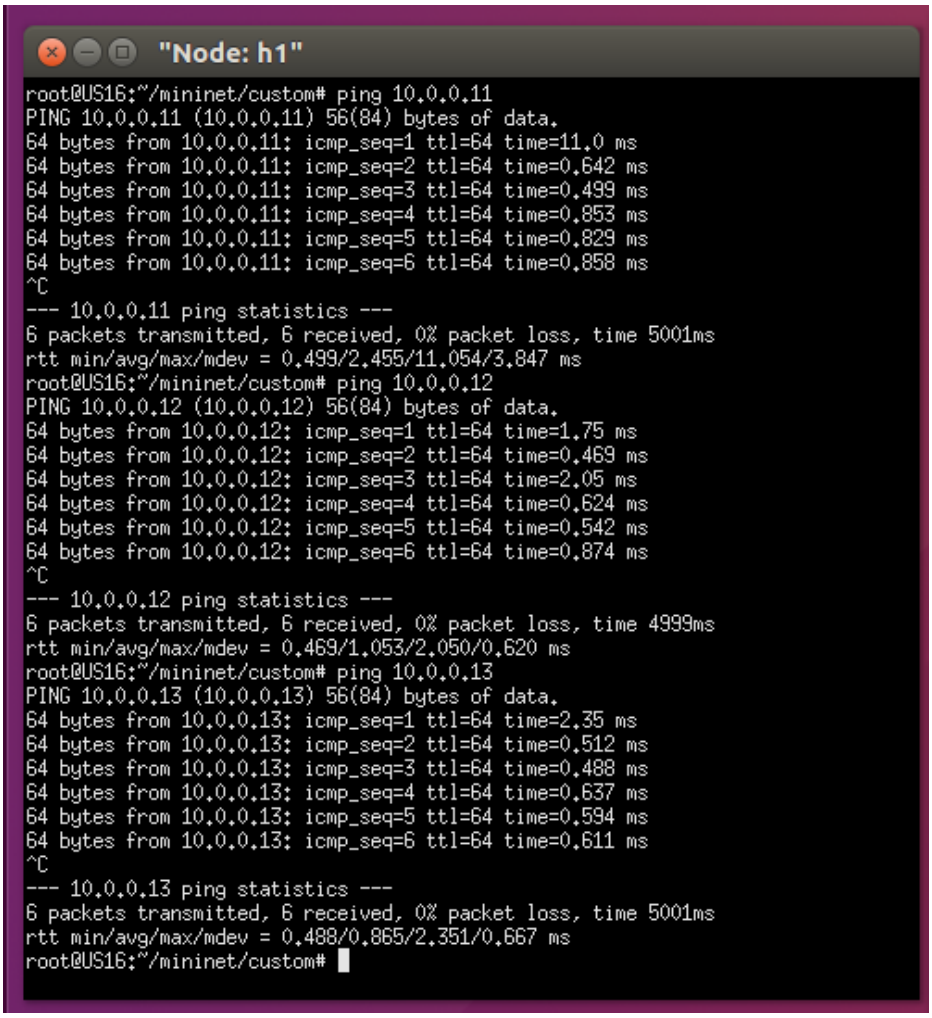

Figura 20. Conectividad desde el nodo1 hacia los servers

TESTEO DE TRÁFICO TCP (BW) Y UDP (JITTER)

En esta etapa es donde realizaremos pruebas de tráfico, se usarán los protocolos más utilizados para medir el rendimiento en términos de tráfico. IPERF nos permitió generar flujos de datos y de esta manera obtener el ancho de banda que se utilizó, el traslado de datos, el jitter y el tiempo utilizado.

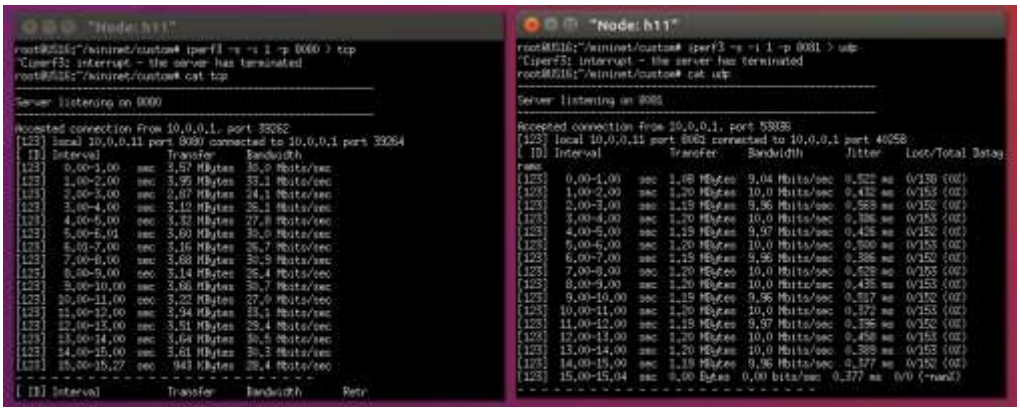

Figura 21. Pruebas IPERF TCP y UDP host cliente

DISPONIBILIDAD Y CAPACIDAD

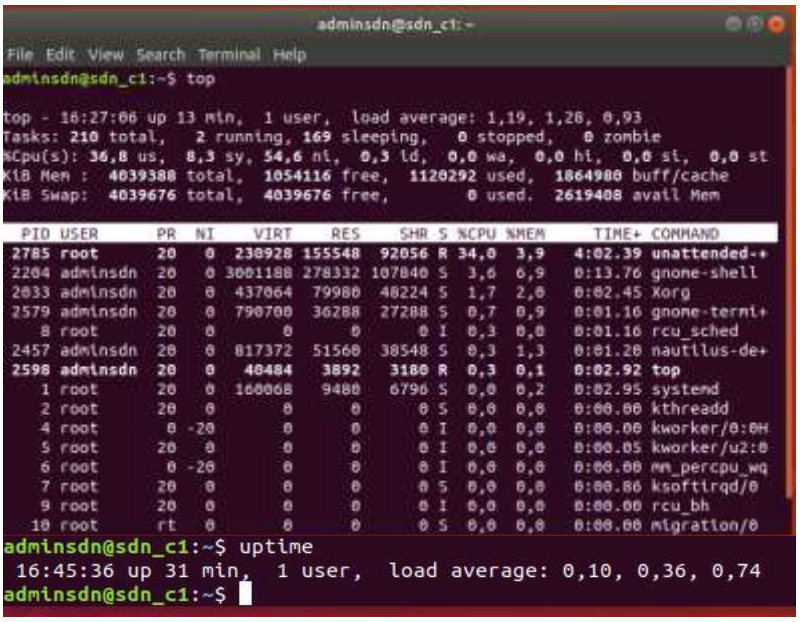

Figura 22. Comando uptime (tiempo de actividad) en ODL

RENDIMIENTO Y ESCALABILIDAD

Cbench, en modo *throughput.* ejecutando el comando:

cbench- c 192.168.1.38 -p 6633 -m 1000 -l 10 -s X -M 100

cbench- c 192.168.1.38 -p 6633 -m 1000 -l 10 -s X -M 100 –t

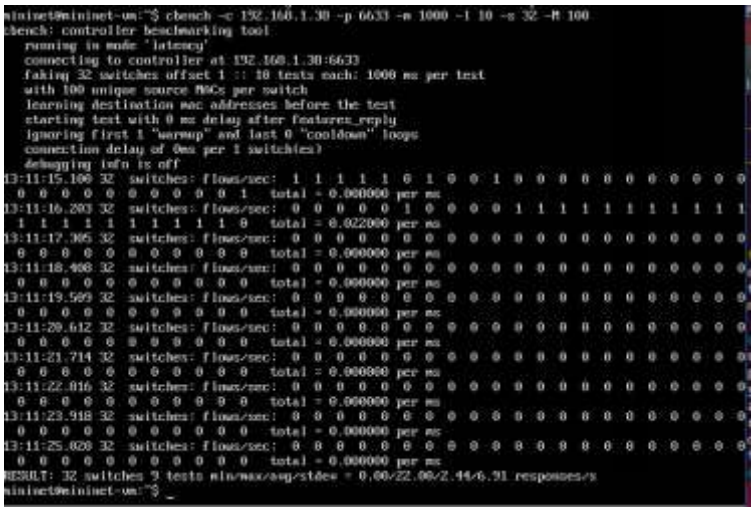

Figura 23. Cbench

1.1 Resultados en Tablas y Figuras

Las pruebas se ejecutaron utilizando el comando IPERF y se generaron los gráficos utilizando la herramienta Gnuplot, obteniendo los siguientes resultados:

Tabla 9. *TRÁFICO TCP - BW / Mbps - HACIA EL HOST H11*

NOMBRE DEL HOST (NODO)	TRÁFICO TCP - BW / Mbps - HACIA EL HOST H11														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
H1	30.0	33.1	24.1	26.1	27.8	30.0	26.7	30.9	26.4	30.7	27.0	33.1	29.4	30.5	30.3
H2	48.1	37.3	38.8	44.0	31.0	40.2	44.9	47.6	44.9	29.3	44.7	45.7	45.1	43.0	45.3
H3	45.6	40.5	43.4	45.4	44.5	42.4	42.3	48.1	42.2	46.2	43.7	46.2	43.8	40.7	47.8
H4	46.6	46.7	45.0	44.2	44.1	46.7	49.1	40.5	34.6	44.6	42.9	45.9	43.5	41.6	49.0
H5	42.9	36.7	40.1	44.8	36.0	43.9	48.2	48.5	47.3	44.8	45.6	45.8	47.1	48.4	45.7
H6	46.3	41.3	40.8	41.9	41.3	41.6	47.5	45.1	42.3	47.7	48.1	46.3	44.9	42.6	46.8
H7	45.3	39.5	46.9	44.8	37.4	42.7	45.7	47.6	41.2	48.5	45.6	47.5	47.3	47.0	48.3
H8	42.6	38.5	45.3	44.2	30.4	44.3	43.7	46.5	46.7	42.8	46.7	45.0	46.8	47.6	46.2
H9	40.8	38.6	43.3	44.3	32.7	40.0	46.3	45.3	43.1	43.4	45.0	46.5	46.0	46.6	47.2
H10	45.1	37.6	38.7	45.6	30.4	25.9	44.6	41.7	45.7	42.4	36.9	34.2	35.1	44.8	40.5
PROMEDIO	43.330	38.980	40.640	42.530	35.560	39.770	43.900	44.180	41.440	42.040	42.620	43.620	42.900	43.280	44.710

Tabla 10. *TRÁFICO UDP - Jiiter / msec - HACIA EL HOST H11*

NOMBRE DEL HOST (NODO)	TRÁFICO UDP - Jiiter / msec - HACIA EL HOST H11														
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
H1	0.522	0.432	0.569	0.386	0.426	0.500	0.386	0.528	0.435	0.517	0.372	0.396	0.458	0.389	0.377
H2	0.412	0.457	0.404	0.484	0.568	0.452	0.402	0.568	0.500	0.511	0.417	0.445	0.416	0.709	0.198
H3	0.367	1.058	0.376	0.331	0.371	0.339	0.336	0.470	0.458	0.538	0.431	0.435	0.386	0.388	0.932
H4	0.535	0.379	0.446	0.400	0.380	0.404	0.494	1.128	0.416	0.484	0.394	0.462	0.511	0.377	0.395
H5	0.439	0.190	0.477	0.566	0.348	0.415	0.370	0.566	0.333	0.388	0.440	0.193	0.395	0.366	0.473
H6	0.673	0.529	0.424	0.403	0.472	0.479	0.890	0.437	0.650	0.358	0.470	0.406	0.381	0.366	0.332
H7	0.467	0.743	0.828	0.965	0.340	0.564	0.651	0.470	0.386	0.397	0.376	0.372	0.520	0.369	0.374
H8	0.463	0.514	0.394	0.384	0.513	0.420	0.456	0.486	0.256	0.498	0.481	0.373	0.396	0.387	0.527
H9	0.536	0.393	0.783	0.467	0.596	1.656	0.290	0.368	0.385	0.431	0.399	0.359	0.472	0.472	0.464
H10	0.394	0.472	0.481	0.399	0.498	0.399	0.478	0.479	0.376	0.480	0.402	0.484	0.480	0.396	0.380
PROMEDIO	0.481	0.517	0.518	0.479	0.451	0.563	0.475	0.550	0.420	0.460	0.418	0.393	0.442	0.422	0.445

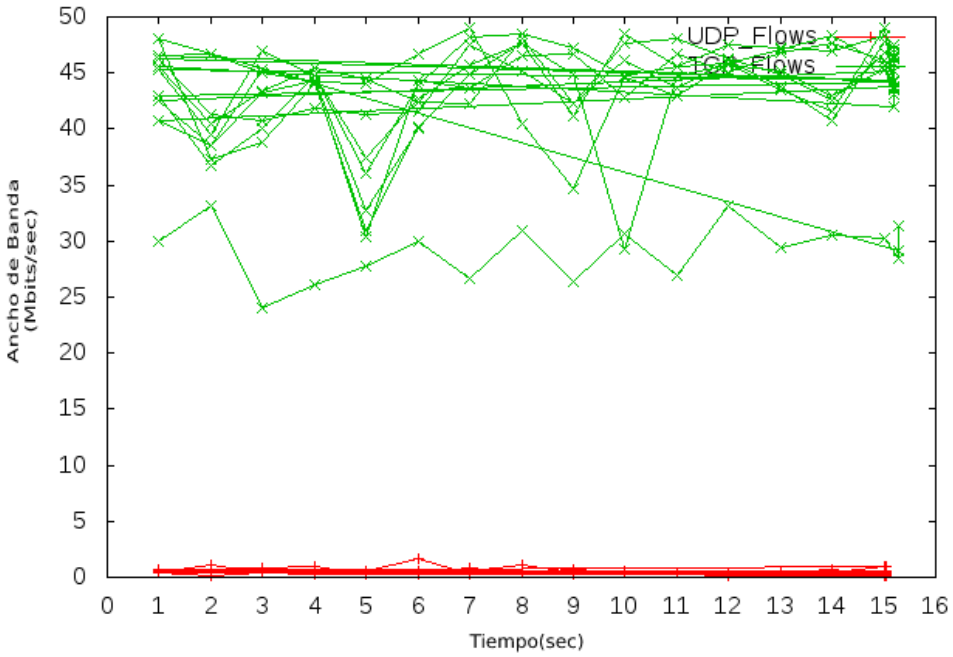

Figura 24. TRÁFICO TCP - BW / Mbps - UDP - Jiiter / msec - HACIA EL HOST H12

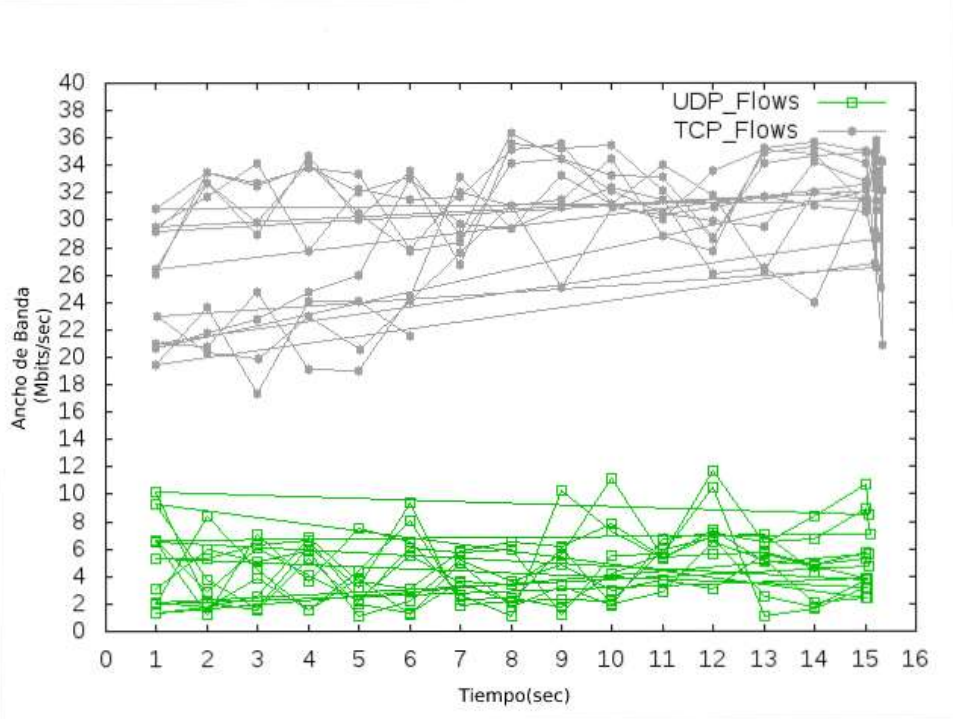

Figura 25. TRÁFICO TCP - BW / Mbps - UDP - Jiiter / msec - HACIA EL HOST H13

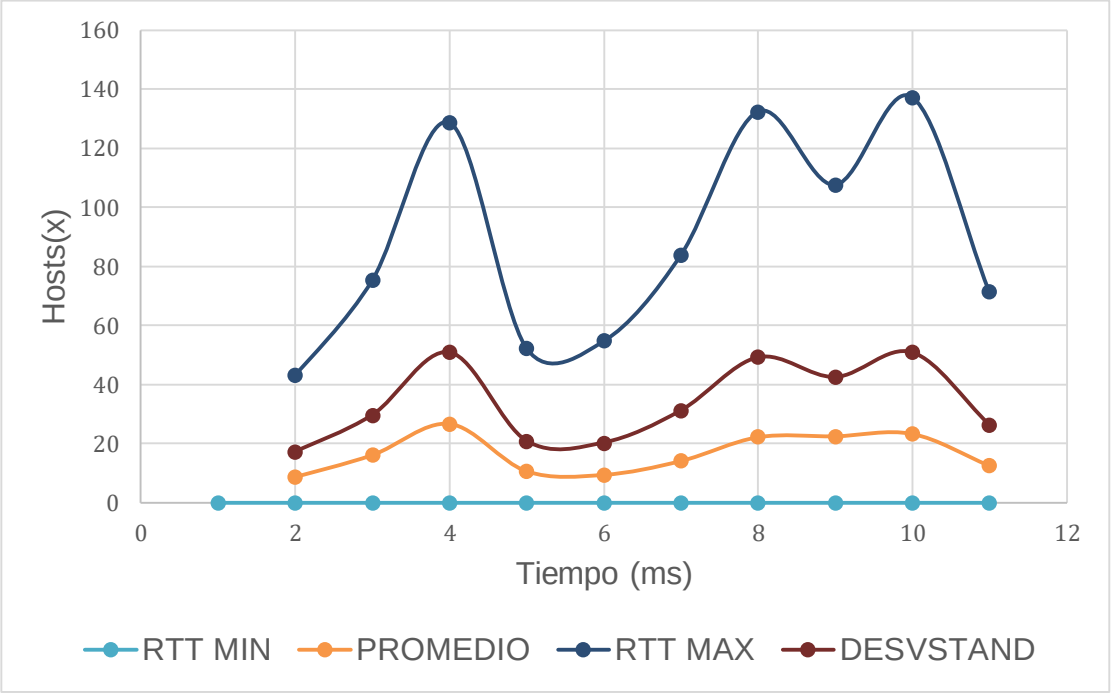

Figura 26. Tráfico ONOS - ICMP – Hacia el H11

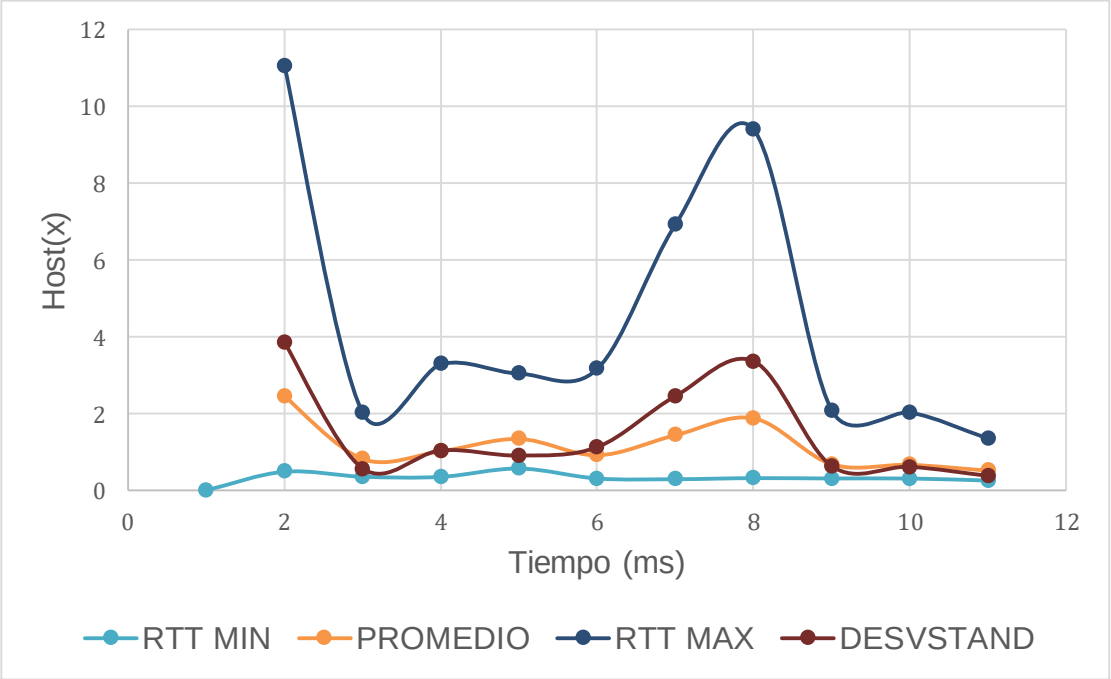

Figura 27. Tráfico ODL - ICMP – Hacia el H11

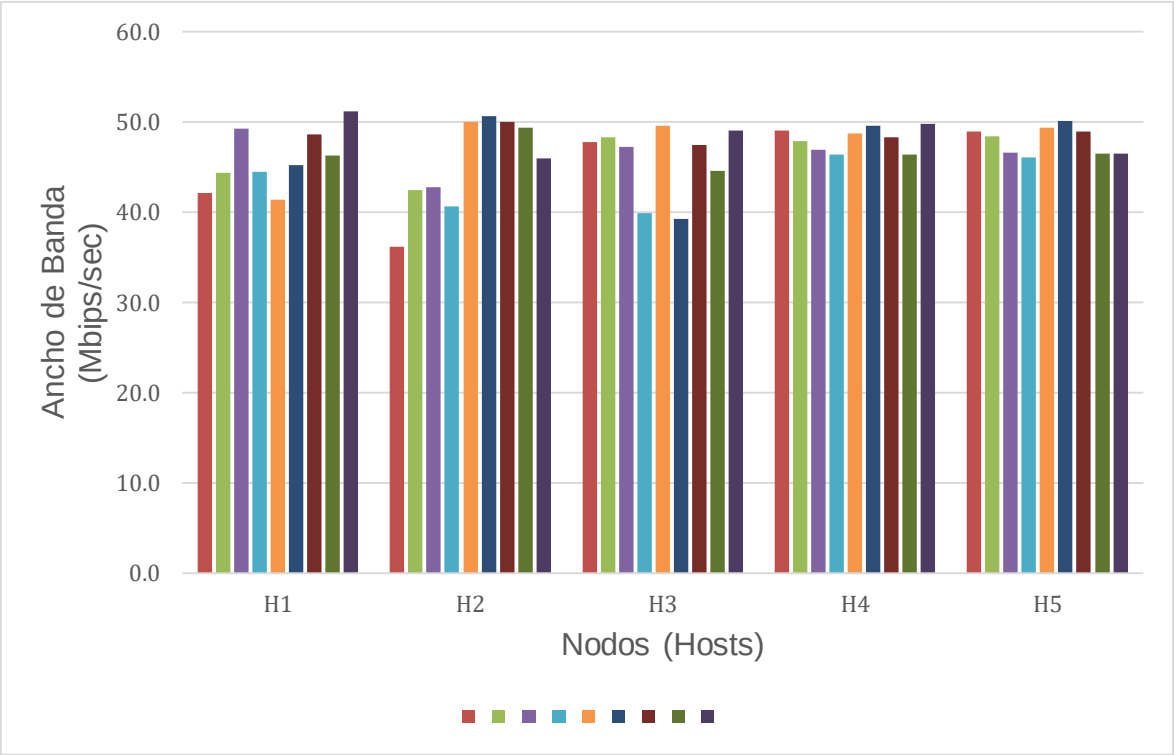

Figura 29. Tráfico TCP - ONOS - BW / Mbps - HACIA EL HOST H12

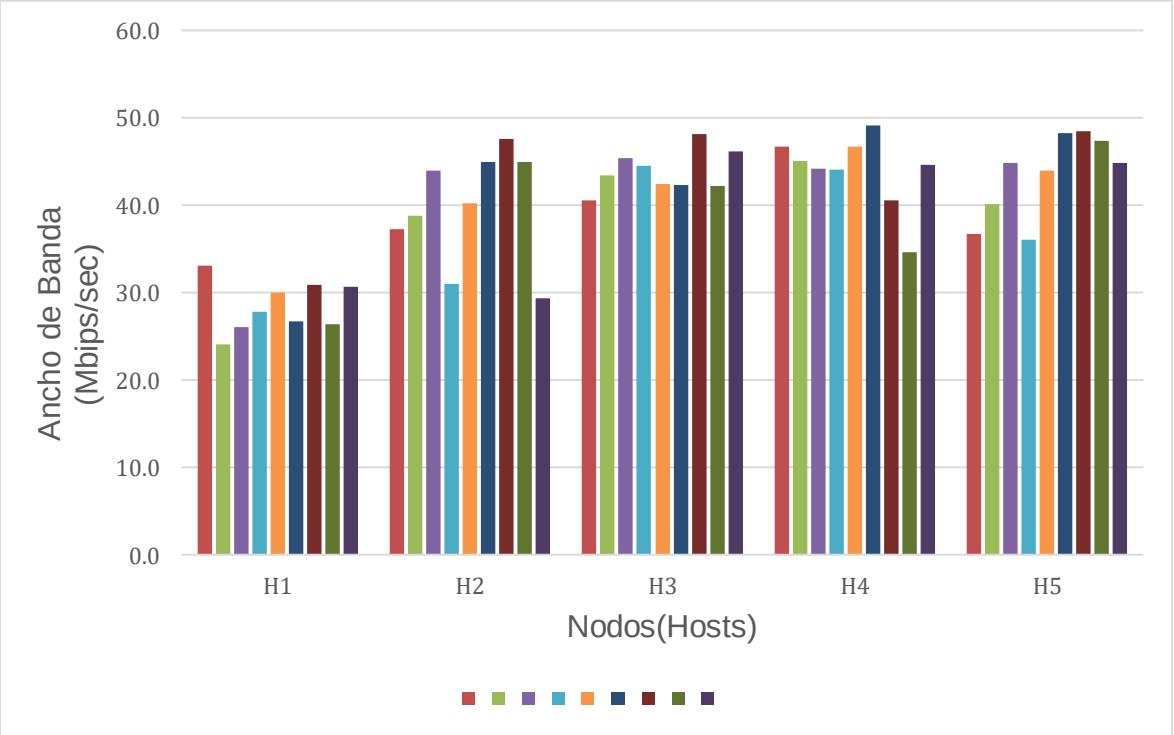

Figura 28. Tráfico TCP – ODL - BW / Mbps - HACIA EL HOST H11

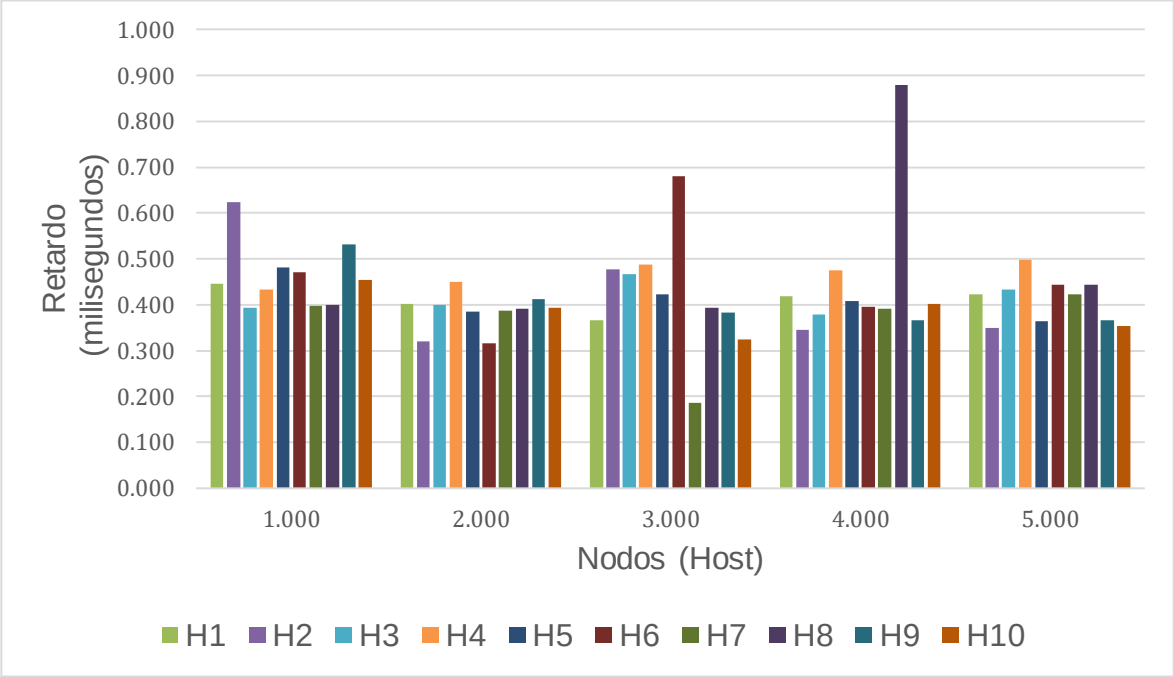

Figura 30. Tráfico UDP – ONOS – Retardo / msec - HACIA EL HOST H12

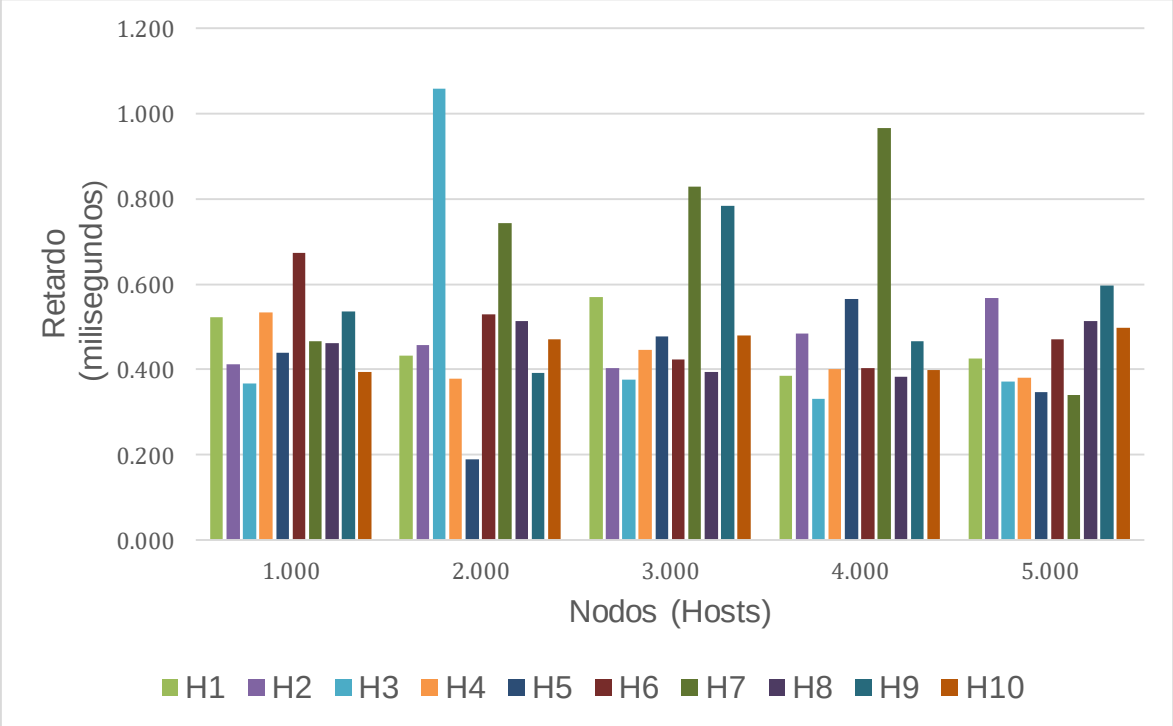

Figura 31. Tráfico UDP – ONOS – Retardo / msec - HACIA EL HOST H11

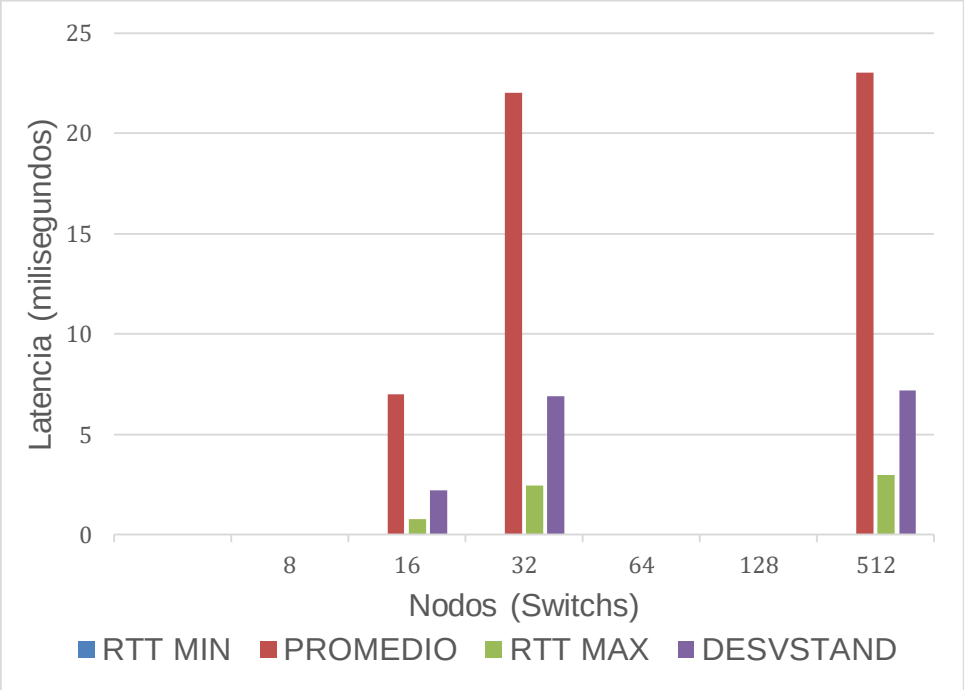

Figura 32. Latencia ODL

.

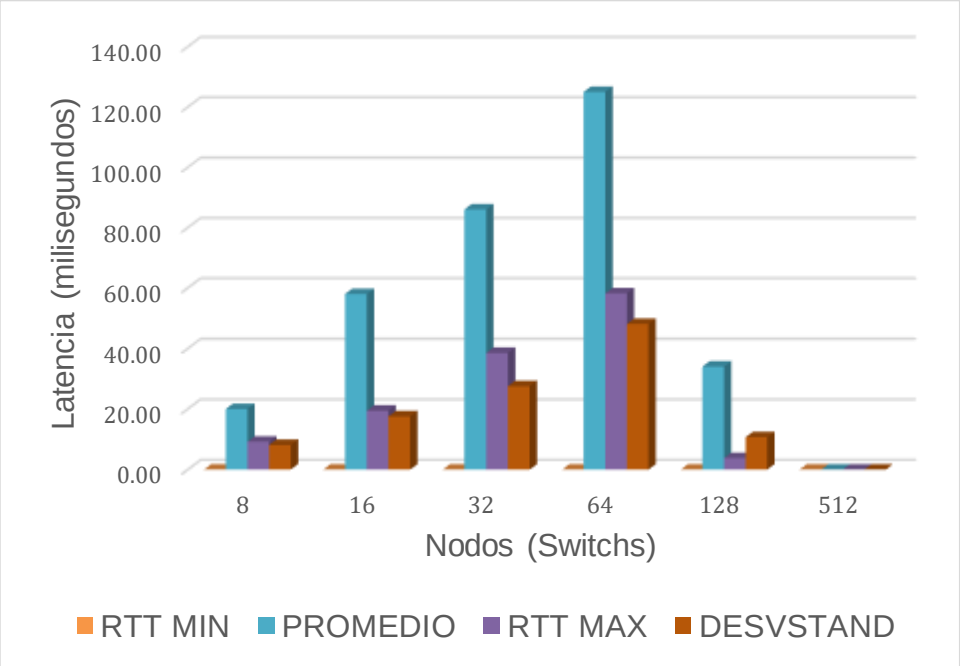

Figura 33. Latencia ONOS

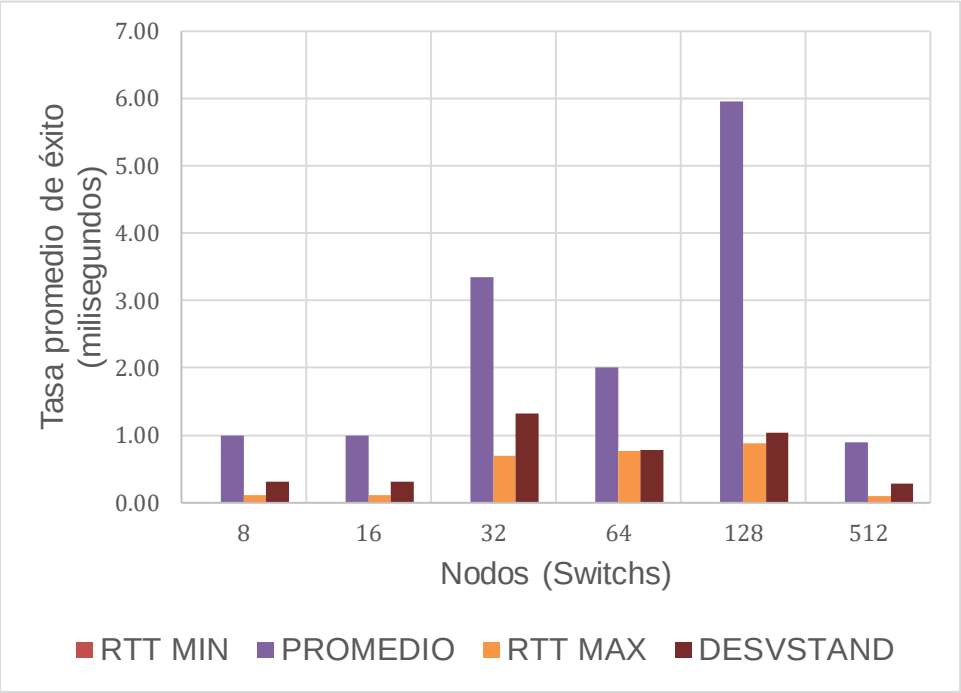

Figura 34. Tasa promedio de éxito ODL

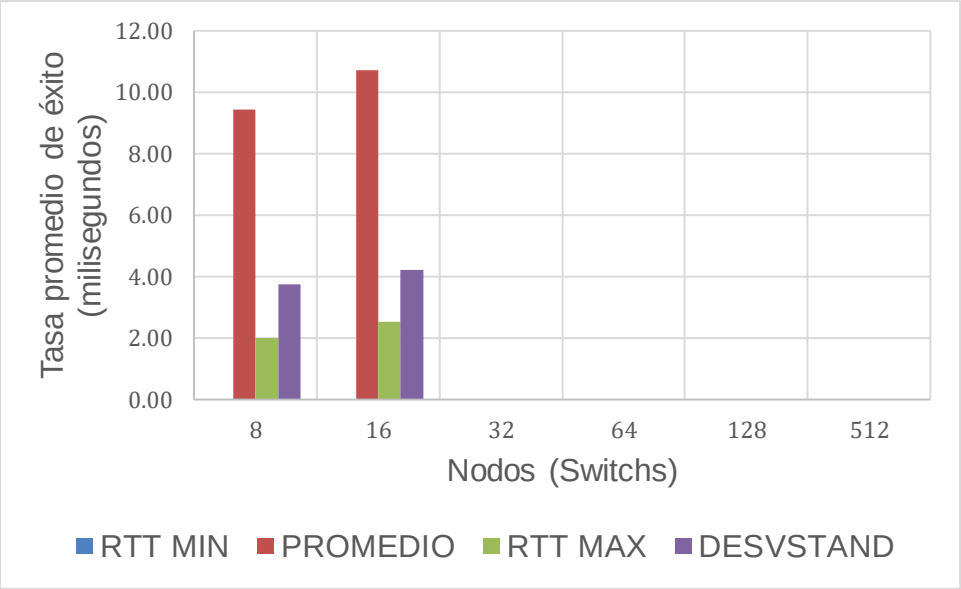

Figura 35. Tasa promedio de éxito ONOS

Tabla 11. *Resultados Versus*

CONSOLIDADO DE RESULTADOS				
	ODL	**CUMPLE**	**ONOS**	**CUMPLE**
RTT MIN	**0.357**		**0.063**	**X**
RTT MAX	**2.040**	**X**	**94.594**	
TCP-BW	**41.237**		**46.463**	**X**
UDP-JITTER	**0.491**		**0.410**	**X**
LATENCIA	**18.623**	**X**	**94.040**	
THROUGHPUT	**3.483**		**16.330**	**X**

Según el consolidado el controlador ONOS obtiene mejores resultados, confirmando la hipótesis del presente trabajo.

1.2 Discusión de resultados

Como parte del proyecto esta sintetizar los resultados, pero al no tener antecedentes sólidos para ejecutar las pruebas, implico que tendríamos que evaluar desde el inicio cual es la forma óptima y correcta de instalar un controlador y que sea operativo.

Según lo observado, detectamos que el tiempo de respuesta es mayor en el primer paquete de respuesta enviado al controlador, esto se debe a que el conmutador haciendo uso del protocolo OpenFlow no cuenta con registros que le sirvan de comparación o base de conocimiento, entonces demora en agregar las tablas de flujos respectivamente.

Posteriormente, el paquete es enviado con un mensaje ARP con un mensaje PACKET IN y lo propaga hasta llegar al switch de acceso. Las pruebas generaron resultados interesantes que permitieron analizar aspectos importantes como el rendimiento y la capacidad de adaptación del controlador SDN.

1.3 Aporte práctico (propuesta, si el caso lo amerita)

La presente investigación tiene como referencia la revisión bibliográfica concerniente al estudio de las Redes Definidas por Software, buscando obtener información que acredite las ventajas de la implementación de la red programable En virtud de ese trabajo previo se encontró las principales características para la selección de controladores SDN.

Compatibilidad multiplataforma: Al elegir un controlador, tenemos que tomar en cuenta el soporte de plataforma multihilo, que afecta el rendimiento del controlador y velocidad de desarrollo. Python, C ++, y Java son los idiomas más utilizados para los controladores SDN. En general, los controladores codificados en Java tienen la característica para ejecutar multiplataforma y presenta aceptable modularidad, los controladores codificados en C proporcionan un alto rendimiento, pero carecen de gran modularidad, una buena gestión de memoria y GUI amigables para el usuario, y por último los codificados en Python carecen del real manejo multi-threading.

Interfaces hacia el sur API: Son aquellas que permiten el control sobre la red. El controlador usa las API para realizar cambios dinámicos en reglas de reenvío instaladas en los dispositivos del plano de datos que consisten de: interruptores, enrutadores, etc. OpenFlow es el protocolo más utilizado para la comunicación entre las API hacia el sur y el controlador SDN.

Interfaces hacia el norte: Las API en dirección norte son utilizadas por la capa de aplicación para comunicarse con el controlador. Ellos son la parte más crítica en la arquitectura del controlador SDN. El beneficio más valioso de SDN se deriva de su capacidad para apoyar y habilitar aplicaciones innovadoras. Porque son tan críticos, las API hacia el norte deben admitir una amplia variedad de aplicaciones. Estas API también

permiten la conexión con pilas automatizadas como OpenStack o CloudStack utilizados para la administración de la nube. En la actualidad, el protocolo de transferencia (REST) parece ser el más utilizado la interfaz hacia el norte y la mayoría de los controladores lo implementan.

Soporte de OpenFlow: El protocolo OpenFlow es un habilitador clave para el software definido DE redes. Fue uno de los primeros protocolos que se usaron para la comunicación hacia las interfaces. Otorga un fácil manejo en el plano de reenvío de los interruptores OpenFlow. Al elegir un Controlador OpenFlow, tenemos que entender la funcionalidad de OpenFlow que el controlador admite, así como la hoja de ruta de desarrollo implementar versiones más nuevas de OpenFlow, como v1.3 o v1.4. Una razón para tener que tomar esto en consideración es esa funcionalidad importante como el soporte de IPv6, por ejemplo, no es parte de OpenFlow v1.0, pero es parte de OpenFlow v1.3 estándar.

Programabilidad de red: Es el beneficio más importante de la introducción SDN para hacer frente a la sin precedentes complejidades de gestión en la red de hoy con la explosión en la cantidad de dispositivos conectados y despliegue de nuevos servicios. El paradigma definido por software viene a introducir la automatización y la dinámica en el proceso de gestión. Los scripts automatizados se pueden ejecutar a través interfaces de línea de comando (CLI) y aplicaciones pueden ser desplegado en la parte superior de la plataforma del controlador para realizar tareas predefinidas y funciones de gestión. El soporte de la programabilidad de la red se basa esencialmente en su grado de integración de una gran cantidad de direcciones hacia las norte interfaces, una buena interfaz gráfica de usuario y una CLI.

Eficiencia (Rendimiento, Fiabilidad, Escalabilidad y Seguridad) La eficiencia del controlador es un término general usado para referirse a los diferentes parámetros - rendimiento, escalabilidad, confiabilidad y

seguridad. Varias métricas, como el número de interfaces que el controlador puede manejar, latencia y rendimiento. Del mismo modo, hay varias métricas que definen la escalabilidad, confiabilidad y seguridad.

Respaldo de la Asociación: La experiencia en la red, dominios de la computadora, y la capacidad económica de los socios de la organización son los principales criterios que desvían la confianza y el uso de productos. Cisco, Linux Foundation, Intel, IBM, Juniper, etc. son ejemplos de organizaciones acreditadas que ingresan al mercado de SDN. y participar en el desarrollo de controladores. A continuación, se expresa en la Tabla 2, otras características que se han considerado.

Tabla 12. *Características para la selección de controladores SDN*

	ONOS	**Open DayLight**	**NOX**	**POX**	**RYU**	**Beacon**	**Maestro**	**Flood-Light**	**Iris**	**MUL**	**Runos**	**LibFluid**
Programming Language	Java	Java	C++	Python	Python	Java	Java	Java	Java	C	C++	C++
GUI	Web Based	Web Based	Python + QT4	Python + QT4	Yes	Web Based		Web Java Based	Web Based	Web Based	Web Based	
Documentation	Good	Very Good	Poor	Poor	Fair	Fair	Poor	Good	Fair	Fair	Fair	Fair
Modularity	High	High	Low	Low	Fair	Fair	Fair	Fair	Fair	Fair	Fair	Fair
Distributed/ Centralized	D	D	C	C	C	C	C	C	C	C	D	-
Platform Support	Linux, MAC OS, And Windows	Linux, MAC OS, And Windows	Most Supported On Linux	Linux, MAC OS, And Windows	Most Supported On Linux	Linux, MAC OS, And Windows	Linux, MAC OS, And Windows	Linux, MAC OS, And Windows	Linux, MAC OS, And Windows	Most Supported On Linux	Most Supported On Linux	Most Supported On Linux
Southbound APIs	OF1.0, 1.3, NETCONF	OF1.0, 1.3, 1.4, NETCONF/ YANG, OVSDB, PCEP, BGP/LS, LISP, SNMP	OF 1.0	OF 1.0	OF 1.0, 1.2, 1.3, 1.4, NETCONF, OFCONFIG	OF 1.0	OF 1.0	OF 1.0 , 1.3	OF 1.0, 1.3, OVSDB	OF 1.4, 1.3, 1.0, OVSDB, OFCONFIG	OF 1.3	OF 1.0, 1.3

	ONOS	Open DayLight	NOX	POX	RYU	Beacon	Maestro	Flood-Light	Iris	MUL	Runos	LibFluid
Northbound APIs	REST API	REST API	REST API	REST API	REST API	REST API	REST API	REST API	REST API	REST API	REST API	
Partner	ON.LAB, AT&T, Ciena, Cisco, Ericsson, Fujitsu, Huawei, Intel, Nec, Nsf.Ntt Comunnication, Sk Telecom	Linux Foundation With Memberships Covering Over 40 Companies, Such As Cisco, IBM, NEC	Nicira	Nicira	Nippo Telegraph And Telephone Corporation	Stanford University	RICE, NSF	Big Switch Networks	ETRI	Kulcloud	ARCCN	ONF
Multithreading Support	Y	Y	NOX_MT	N	Y	Y	Y	Y	Y	Y	Y	Y
OpenStack Support	N	Y	N	N	Y	N	N	N	N	Y	N	N
Application Domain	Datacenter, WAN and Transport	Datacenter	Campus	Campus	Campus	Research	Research	Campus	Carrier-Grade	Datacenter	WAN, Telecom and Datacenter	

Según la descripción de la Tabla ONOS y OpenDaylight son los controladores más destacados. Estos dos controladores están codificados en Java y ejecutan plataformas cruzadas y presentan alta modularidad, usando el contenedor OSGI que permite carga de paquetes en tiempo de ejecución. Se puede decir que han heredado el poder de Java / JavaScript en la programación gráfica de interfaces de usuario,

El controlador ONOS está diseñado principalmente para el operador redes. Les da la capacidad de proporcionar nuevos servicios SDN junto con sus servicios patentados iniciales. La arquitectura ONOS está diseñado para mantener redes de alta velocidad y de gran escala. Sus principales características distintivas es su apoyo para redes híbridas.

Sin embargo, OpenDaylight se centra principalmente en los centros de datos. Muchas interfaces hacia el sur se han agregado (HTTP, COAP, PCEP, LACP, OpFlex, SNMP, etc.) y se han implementado nuevos módulos (IoT Data Broker (IoTDM), canal seguro unificado de USC, etc.). Por lo tanto, es el primer controlador que ingresa al dominio IoT. Compatible con una amplia gama de interfaces hacia el sur y paradigma de control distribuido, parece ser el controlador de la Internet del futuro.

El soporte de OpenStack Neutron plugin en su arquitectura también tiene una importancia notable al implementar bordes definidos por software que responden al borde de proliferación informática Ryu, presentando características justas, es una buena opción para pequeños negocios y aplicaciones de investigación. Al estar codificado en Python, esto controlador presenta instalaciones para aplicaciones y módulos para desarrollo. Sin embargo, su falta de alta modularidad y su la imposibilidad de ejecutar multiplataforma limita su uso amplio en el mercado real y aplicaciones.

Luego, de definir los controladores para el entorno de pruebas se recurre a las tecnologías de virtualización existentes, siendo la elegida Oracle Virtual Box v6.1 por ser una solución gratuita y de código abierto.

Nuestro principal recurso fue una laptop con las funciones de anfitrión con las siguientes características:

Tabla 13. *Características del equipo anfitrión*

Tipo	CPU	RAM	HD Sólido	Sistema Operativo	Versión
Laptop	Intel Core I5, con el máximo de núcleos habilitado(4).	12 GB	500 GB	Microsoft Windows 10 Pro	2004

Asimismo, cada máquina virtual tiene las siguientes características.

Tabla 14. *Características de las máquinas virtualizadas*

Máquina Virtual	CPU	RAM	HD	Sistema Operativo	Versión
Mininet	2 núcleos	4 MB	40 GB	Linux Ubuntu 16.04	V2.2.2 con Openflow 1.3
OpenDayLight SODIUM - SR1	2 núcleos	4 MB	40 GB	Linux Ubuntu Server	18.04
ONOS Toucan 2.3.0	2 núcleos	4 MB	40 GB	Linux Ubuntu Server	18.04

Se optó por considerar en la implementación versiones actualizadas en controladores y Sistemas Operativos, debido a que los estudios comparativos se encuentran desfasados e incluso no pueden replicarse, debido a factores que se encuentran durante el despliegue, que mencionaremos más adelante.

Figura 36. Grupo de máquinas virtuales en el Despliegue SDN

Posteriormente se procede al diseño de la topología de red en GNS3.

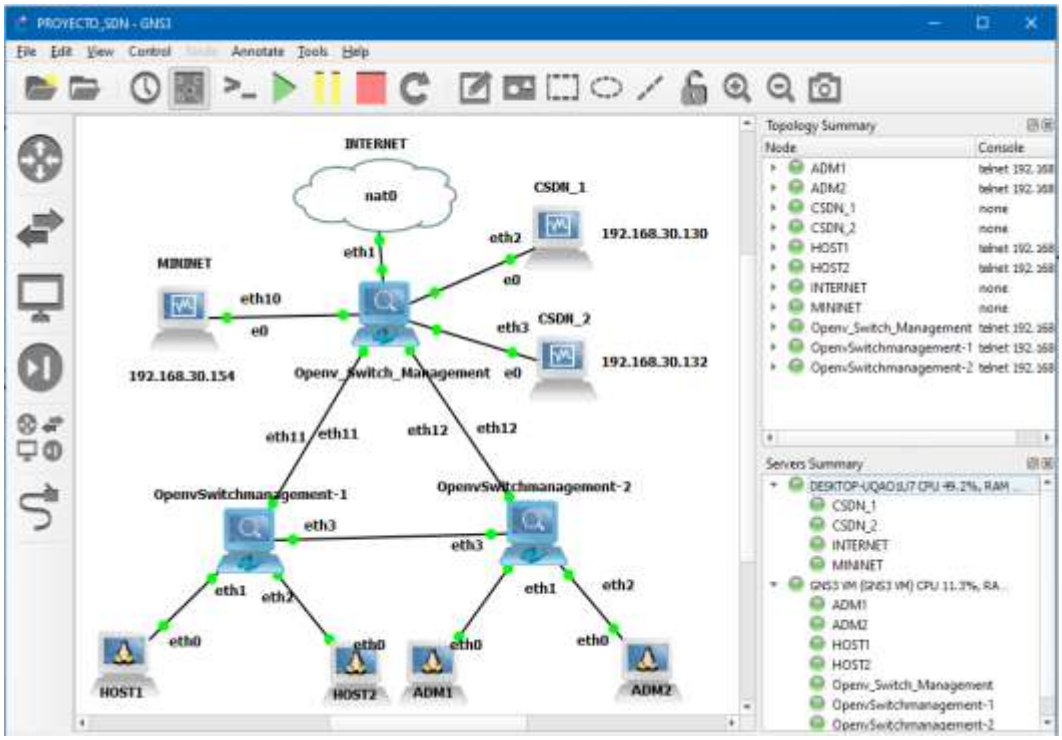

Figura 37. Diseño de la red SDN en GNS3

Llegó el momento de implementar y desplegar el entorno diseñado, y empezaremos por indicar los requerimientos y configuraciones básicas pero fundamentales para el correcto funcionamiento, aquellos que en el arte analizado no se detallan, debemos recordar que estamos ante una arquitectura emergente, que en los próximos años tendrá alta demanda.

Empezamos instalando el Sistema Operativo Linux Ubuntu Server, en las máquinas virtuales, cada componente se está ejecutando sobre un host diferente. La configuración de los adaptadores de red en VirtualBox es esencial, en este caso la administración de la topología la realizo GNS3, por eso en los adaptadores únicamente se le indico en adaptador *"no conectado"*, con el fin de que los IPs se encargue GNS3, por eso aparecerá lo siguiente en la conexión.

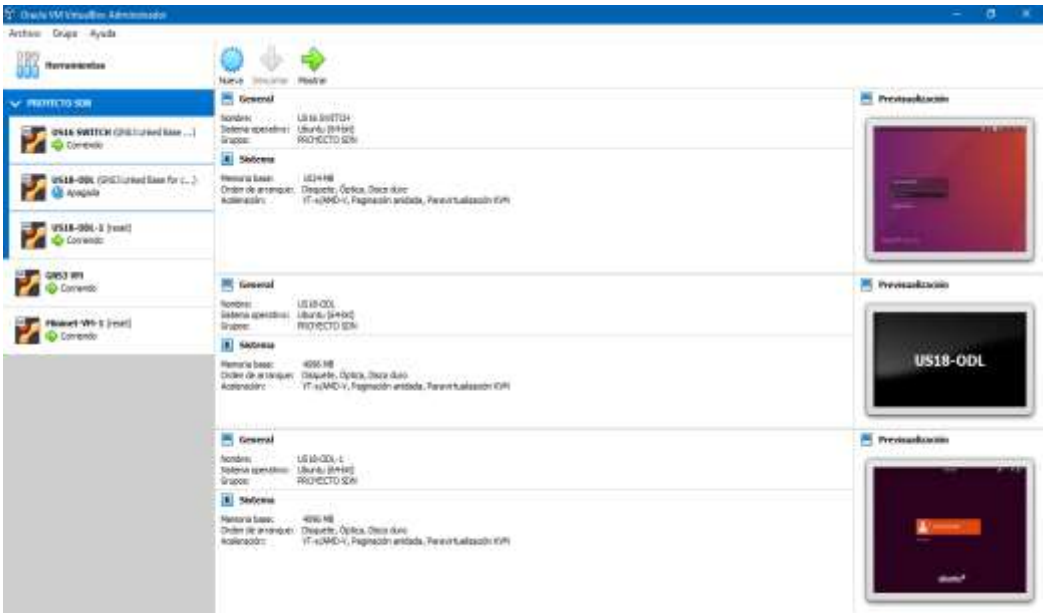

Figura 38. Configuración Adaptador 1

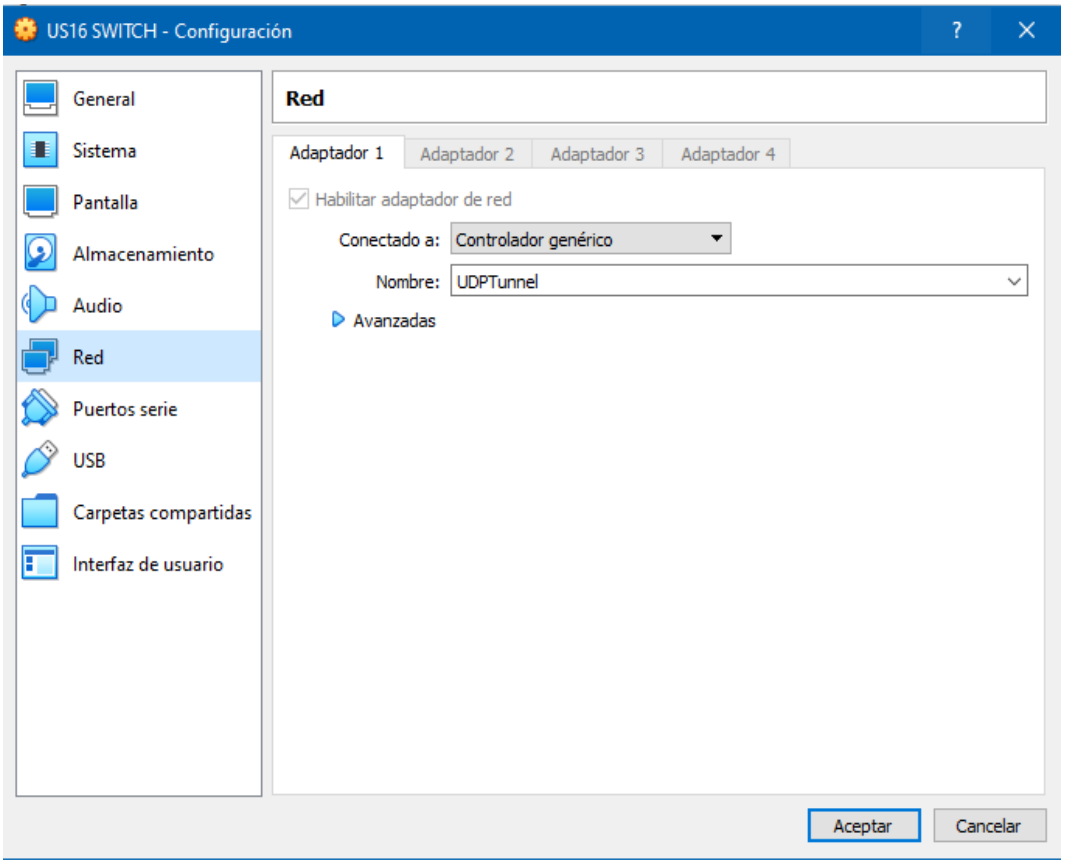

Figura 39. Configuración de Adaptador 3

Para facilitar la instalación de los comandos de la documentación, recomiendo instalar el servicio SSH Port: 22 y habilitar por medio del PUTTY el acceso correspondiente, de igual manera como necesitamos visualizar el funcionamiento del controlador, es necesario instalar una versión GUI de Ubuntu, para que cargue el escritorio y por ene ingresar al navegador FIREFOX.

Asimismo, no todas las versiones de los controladores cuentan con la misma metodología de instalación, por ello se pretende instalar versiones actualizadas de los controladores SDN y generar documentación fidedigna.

Ejecución del Controlador OpenDayLight

cd opendaylight-0.11.1/

./bin/karaf

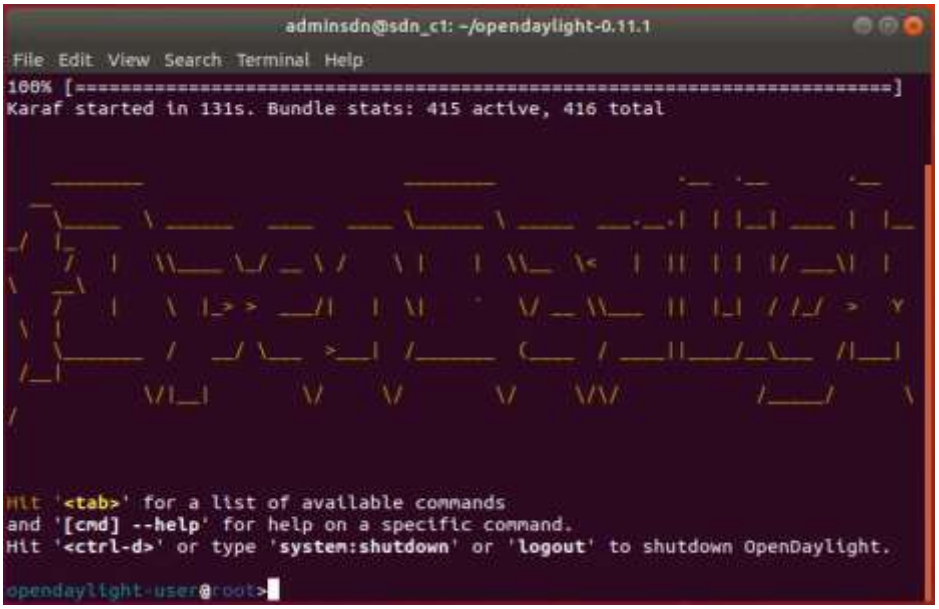

Figura 41. Karaf – Consola OpenDayLight

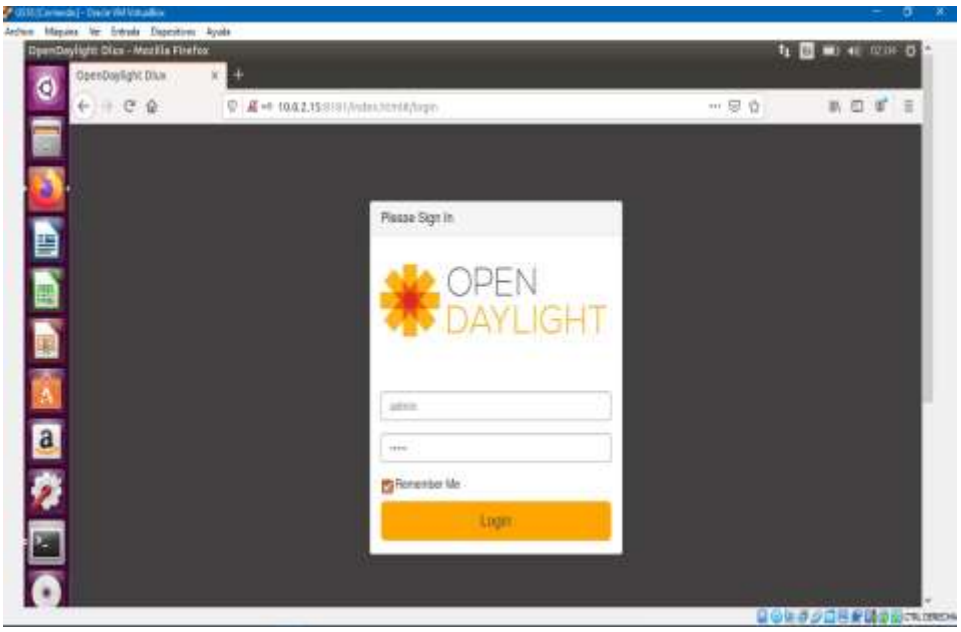

Figura 42. Karaf Portal Logeo - OpenDayLight

Ejecución del CONTROLADOR ONOS

Enter ONOS Gui in firefox

localhost:8181/onos/ui/index.html

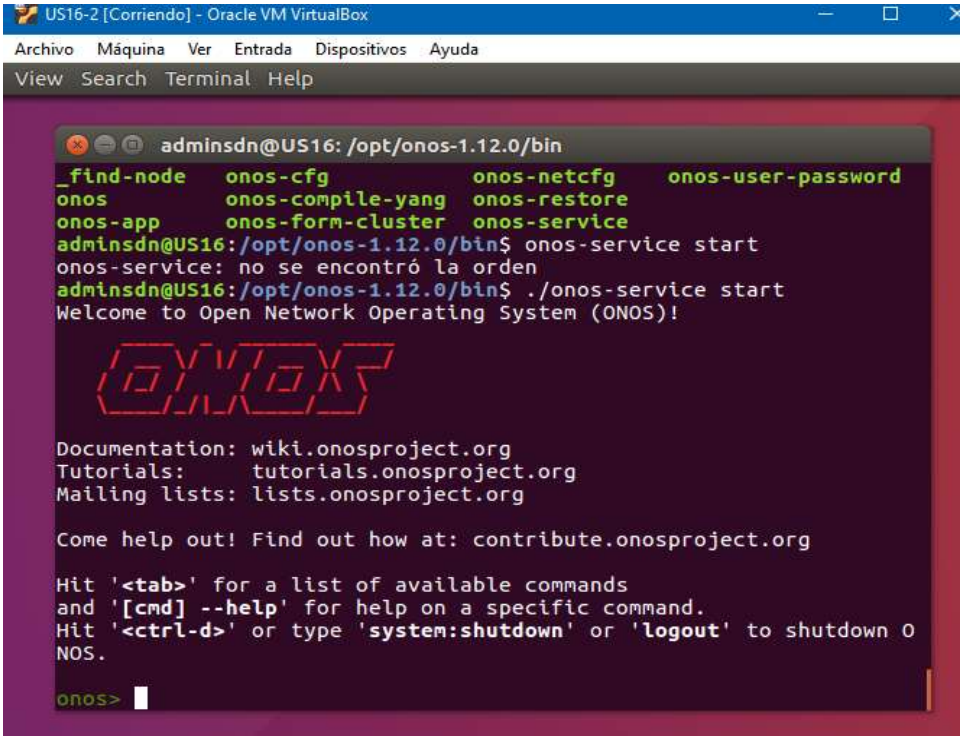

Figura 43. Consola ONOS

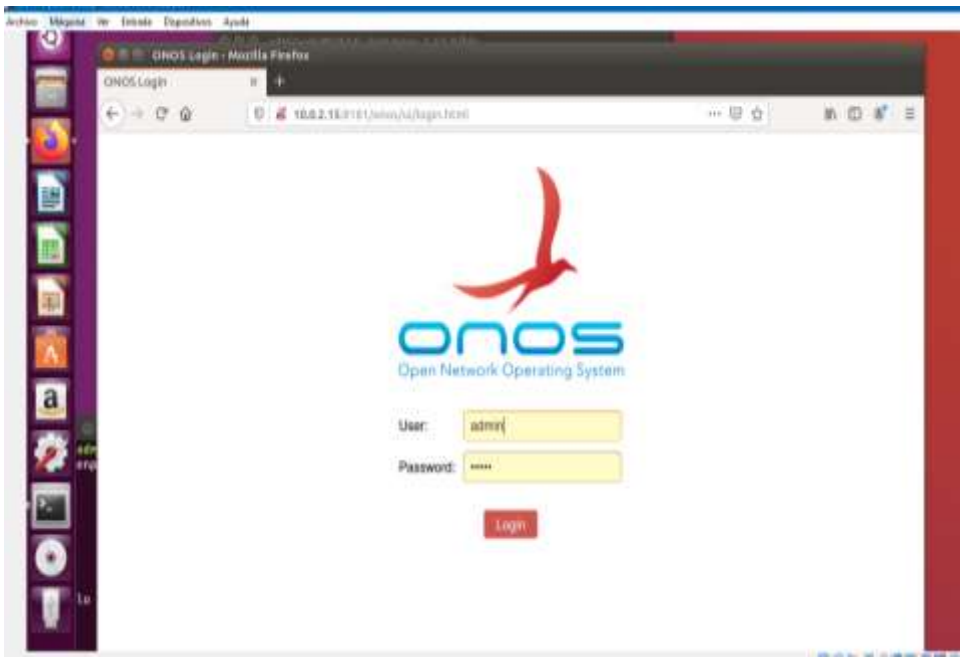

Figura 44. ONOS-Pantalla de Logeo

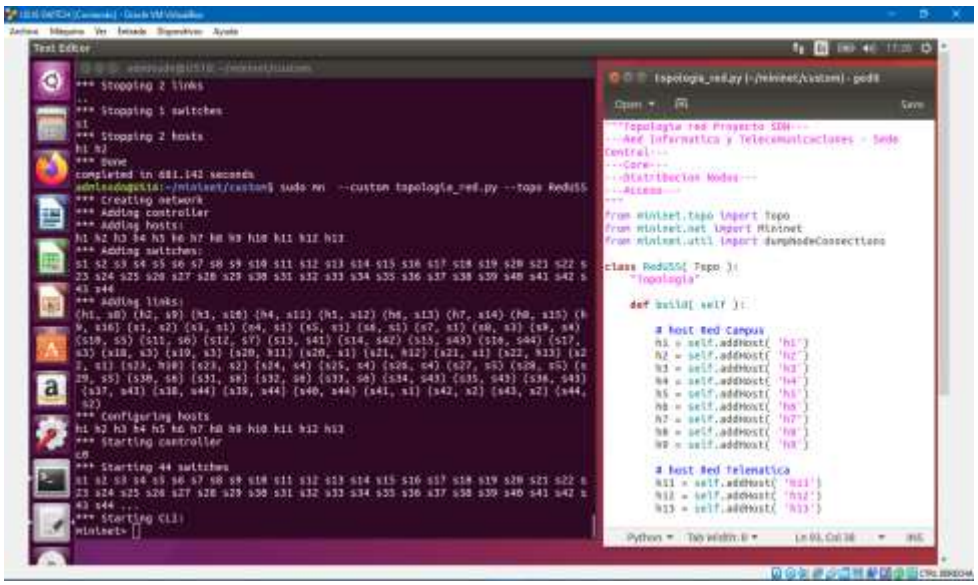

Figura 45. MININET – Topología personalizada

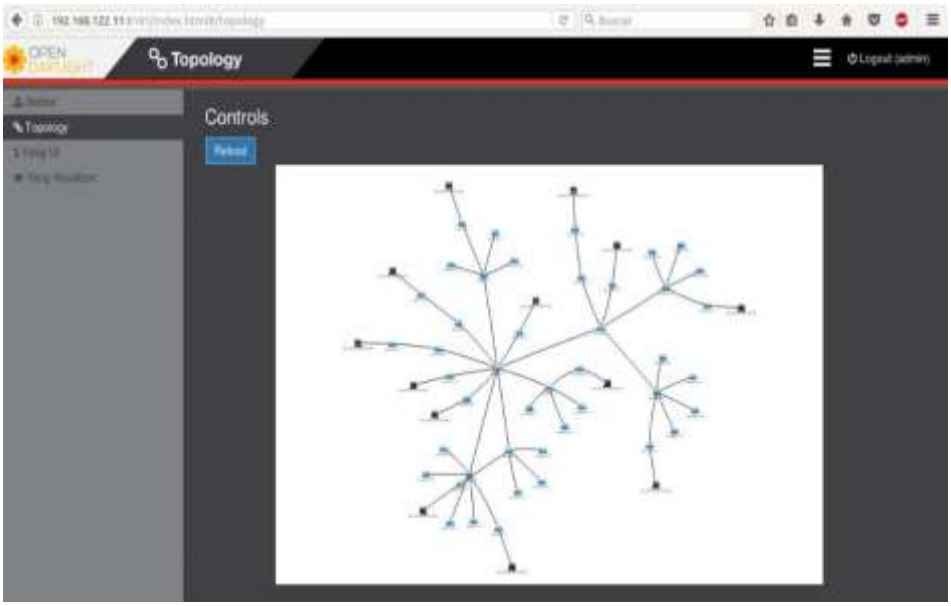

Figura 46. MININET – Topología personalizada - ODL DELUX

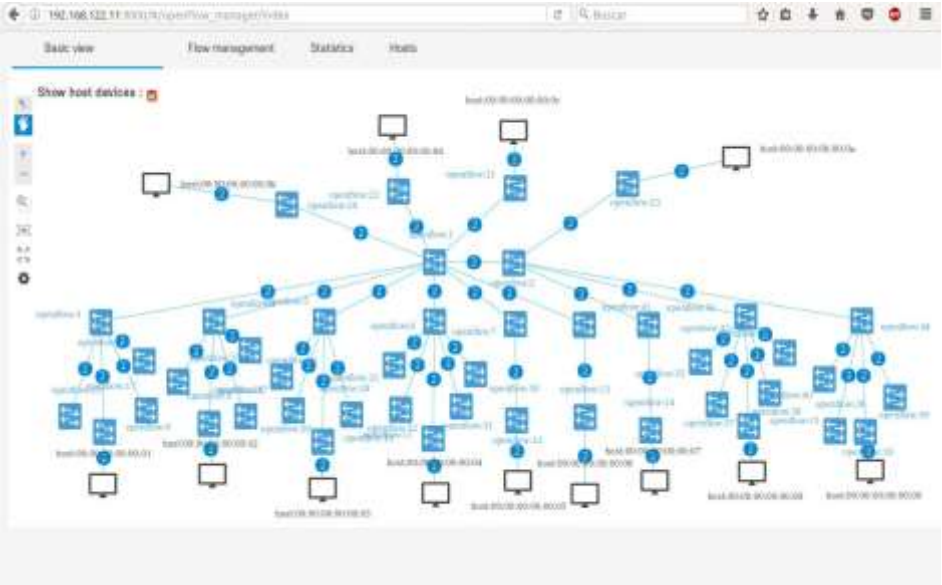

Figura 47. MININET – Topología personalizada – OpenFlowManager OFM

Métricas para evaluación de desempeño.

El acelerado avance en las implementaciones de las Redes Definidas por Software presenta una brecha de conocimientos en el uso de instrumentos tecnológicos que permitan valorizar sus elementos y funciones respectivamente. Uno de los objetivos de esta tarea es evaluar con detenimiento las métricas detalladas por la ONF en un despliegue simulado SDN.

Existen diversas métricas válidas para las SDN, entre las cuales podemos mencionar: las que ha definido la ONF relacionadas con los componentes OpenFlow: *disponibilidad, confiabilidad, capacidad, exactitud, seguridad y rendimiento*.

El complemento Mininet es muy útil, para el análisis de protocolos, es amigable con el usuario por sus interfaces y los comandos que posee para testeo de la red tales como: Wireshark, ping, pingall y Cbench.

Pruebas de configuraciones y desempeño en Mininet

Las pruebas de configuraciones estiman el proceder de las características SDN tales como: conmutación, el establecimiento de los flujos y la comunicación entre el controlador y los elementos de red tales como switches. Los resultados permitirán valuar la capacidad de adecuación con otras tecnologías, utilización de los medios y la confiabilidad del contexto, referenciadas por los indicadores antes mencionados.

IV. CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

En este trabajo se optó por evaluar controladores en sus últimos lanzamientos, la selección de los controladores es una de las etapas importantes en un proyecto debido a que la amplia gama de propuestas en el mercado ofrece novedades, pero a la vez la mayoría no cuentan con el respaldo de despliegues reales, o bajo condiciones de esfuerzo cotidianas. Como parte de nuestro objetivo es realizar el despliegue en una topología real, para ello se personalizo una distribución y lógica cercana a la realidad. El simulador de red Mininet facilito la carga y ejecución del código Python que contiene el diseño antes mencionado. Durante la primera etapa del despliegue la dificultad más notable fue la instalación del Java en su versión 7, que según la guía de instalación requiere. El soporte de Java caducó, por ello no permite instalar desde comandos o vínculos de la consola, se recurrió a repositorios externos.

Luego, de tener éxito en la instalación, se procede a instalar los paquetes adicionales necesarios para la gestión del controlador, adjunto se realizó las pruebas de rendimiento, se proyectó cuadros, figuras y tablas con datos generados por los comandos de análisis. Los resultados otorgaron una visión holística de la arquitectura SDN, logrando comprender el funcionamiento de sus componentes y requisitos.

ODL y ONOS ofrece una variedad de Appilance, que permiten fundamentar el funcionamiento de Appis, las últimas versiones presentan bondades de acuerdo a la vanguardia de la tecnología, como, por ejemplo, la virtualización. Por ende, es importante ejecutar más despliegues híbridos o totales para generar sostenibilidad y base de conocimiento a los futuros proyectos.

4.2. Recomendaciones

Es de vital importancia difundir y desplegar proyectos bajo la Arquitectura SDN, dichas implementaciones aportarán conocimiento y experiencia a los profesionales e investigadores futuros.

Se recomienda utilizar las últimas versiones bajo entornos reales y de alta exigencia para generar experiencias, foros, aportes que permitan una guiada implementación en proyectos reales. Por ejemplo, el uso de dockers o virtualización NFV.

REFERENCIAS.

Aliyu, A., Bull, P. & Abdallah, A. (2017). Implicación del rendimiento y análisis del protocolo OpenFlow SDN.

ALL, O. (2014). SDN architecture. *Palo Alto,, Bayshore Road.* .

BAH, M., Azzouni,A., Nguyen, M. & Pujolle, G. (2019). Evaluación del rendimiento de descubrimiento de topología de los controladores OpenDaylight y ONOS.

Bispo, P., Corujo,D. & Aguiar,R. (2017). Una evaluación cualitativa y cuantitativa de los controladores SDN.

CHAHLAOUI, F. & DAHMOUNI, H. (2018). Hacia redes SDN habilitadas para QoS.

Cisco. (2019). *Reporte sobre tendencias globales en redes 2020. .*

Darianian, M., Williamson, C. & Haque, I. (2017). Evaluación experimental de dos controladores OpenFlow.

Echeverry, M. (2018). Redes de datos definidas por software - SDN, arquitectura, componentes y funcionamiento. *Journal de Ciencia e Ingeniería,*, vol. 10, no. 1, 7.

Fan Yamei, Liao Qing1 & He Qi. (2016). *Investigación y análisis comparativo de la prueba de rendimiento en el controlador SDN.* Beijing, China.

Fancy, C. (2018). Performance Evaluation of SDN controllers POX and Floodlight in Mininet Emulation Environment. *IEEE Xplore*, 5.

Foundation, O. (2014). OpenFlow Switch Specification. .

Foundation, O. N. (s.f.). Obtenido de Open Networking Foundation: https://www.opennetworking.org

Foundation, O. N. (2015). *OpenFlow Switch Specification*. Obtenido de https://www.opennetworking.org/wp-ontent/uploads/2014/10/openflow-switch-v1.5.1.pdf

García, A. (2014). Controladores SDN, elementos para su selección y evaluación. *Revista Telem@tica*, Vol. 13. No. 3,, 11.

Hoang, D. & Pham, M. (2015). Sobre redes definidas por software y el diseño de Controladores SDN.

Jawaharan, R. (2018). Empirical Evaluation of SDN Controllers using Mininet/Wireshark and Comparison with Cbench. *IEEE Xplore*, 2.

KALJIC, E. (2019). A Survey on Data Plane Flexibility and Programmability in Software-Defined Networking. *IEEE Xplore*, 37.

Kaur, K., Kaur, S. & Gupta, V. (2018). Análisis de rendimiento de controladores de flujo abierto basados en python.

Khattak, Z., Awais, M. & Iqbal, A. (2015). Evaluación del rendimiento de OpenDaylight SDN.

LAASSIRI,F., MOUGHIT, M. & IDBOUFKER, N. (2018). *Evaluación de los parámetros de QoS en diferentes arquitecturas SDN usando Omnet 4.6.* Marruecos.

Laissaoui, C. (2016). A measurement of the response times of various OpenFlow/SDN controllers with CBench. *IEEE Xplore*, 2.

Laissaoui, C., Idboufker, N., Elassali, R. & Baamrani, K. (2016). Investigación y análisis comparativo de la prueba de rendimiento en el controlador SDN.

Lee, S. (2019). Performance Comparison of Software Defined Networking Simulators for Tactical Network : Mininet vs. OPNET. *IEEE Xplore*, 6.

List of SDN controller software. (s.f.). Obtenido de https://en.wikipedia.org/wiki/List_of_SDN_controller_software

Mamadou, T. (2019). Topology Discovery Performance Evaluation of OpenDaylight and ONOS Controllers. *IEEE Xplore*, 7.

Mamushiane, L., Lysko, A.& Dlamini, A. (2018). Una evaluación comparativa del rendimiento de los controladores SDN populares.

Nguyen-Ngoc, A. (2018). Benchmarking the ONOS Controller with OFCProbe. *IEEE Xplore*, 6.

ODL. (s.f.). *Opendayligh*. Obtenido de https ://www.opendaylight.org/.

ONOS. (s.f.). *Open Network Operating System (ONOS)*. Obtenido de https ://wiki.onosproject.org/display/ONOS/Wiki+Home.

Ortiz, J. (2016). Evaluation of performance and scalability of Mininet in scenarios with large data centers. *IEEE Xplore,*, 6.

Parra, P. (2015). Redes Definidas por Software: beneficios y riesgos de su implementación en Universidades. *CONAIC*, 7.

Pereira, G., Silva, J. & Sousa, P. (2019). *Estudio comparativo de redes definidas por software.* Braga, Portugal.

Priyadarsini, M., Bera, P. & Bhampal, R. (2018). Análisis de rendimiento de la arquitectura de controlador de red definida por software una encuesta basada en simulación.

Rajaratnam, A. (2017). Software Defined Networks: Comparative Analysis of Topologies with ONOS. *IEEE Xplore*, 5.

Rajaratnam, A., Kadikar, R., Prince, S. & Valarmathi, M. (2017). Redes definidas por software análisis comparativo de topologías con ONOS.

Rowshanrad, S. (2016). Performance evaluation of SDN controllers: Floodlight and OpenDayLight. *IIUM Engineering Journal*, 105.

Rowshanrad, S., Abdi, V. & Keshtgari, M. (2016). Evaluación del Rendimiento de Controladores SDN: Floodlight Y Opendaylight.

Salman,O., Elhajj, I., Kayssi, A. & Chehab, A. (2016). Controladores SDN un estudio comparativo. *In Electrotechnical Conference (MELECON)*, 6.

Sanchez, J. & Espinel, D. (2018). Comparación de tolerancia a fallos de los controladores SDN de ONOS y OpenDaylight.

Shah, S. (2013). An architectural evaluation of SDN controllers. *IEEE Xplore.*, 7.

Shalimov, A. (2013). Advanced study of SDN/OpenFlow controllers. *IEEE Xplore*.

Tootoonchian, A. (2012). On Controller Performance in Software-Defined Networks. *IEEE Xplore.* .

ANEXOS.

Anexo 1. Resolución de aprobación del proyecto de investigación

USS | UNIVERSIDAD SEÑOR DE SIPÁN

FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO
RESOLUCIÓN N°0652-2021/FIAU-USS

Pimentel, 22 de junio de 2021

VISTO:
El Acta de reunión N°1106-2021, remitida mediante oficio N°0236-2021/FIAU-IS-USS del 11 de junio de 2021, para la ejecución de la Tesis: **"ANÁLISIS COMPARATIVO DEL RENDIMIENTO DE CONTROLADORES DE REDES DEFINIDAS POR SOFTWARE DE TIPO OPEN SOURCE PARA LA GESTIÓN DE UNA RED DE COMPUTADORAS"**, presentado por **SANTISTEBAN YNGA BICRY RAUL,** del Programa de estudios de **INGENIERÍA DE SISTEMAS**, y;

CONSIDERANDO:

Que, de conformidad con la Ley Universitaria N° 30220 en su artículo 48° que a letra dice: "La investigación constituye una función esencial y obligatoria de la universidad, que la fomenta y realiza, respondiendo a través de la producción de conocimiento y desarrollo de tecnologías a las necesidades de la sociedad, con especial énfasis en la realidad nacional. Los docentes, estudiantes y graduados participan en la actividad investigadora en su propia institución o en redes de investigación nacional o internacional, creadas por las instituciones universitarias públicas o privadas.";

Que, de conformidad con el Reglamento de grados y títulos en su artículo 21° señala: "Los temas de trabajo de investigación, trabajo académico y *tesis* son *aprobados por el Comité de Investigación* y derivados a la facultad o Escuela de Posgrado, según corresponda, para la emisión de la resolución respectiva. El *periodo de vigencia de los mismos será de dos años*, a partir de su aprobación. En caso un tema perdiera vigencia, el Comité de Investigación evaluará la ampliación de la misma.

Que, de conformidad con el Reglamento de grados y títulos en su artículo 24° señala: La tesis es un estudio que debe denotar rigurosidad metodológica, originalidad, relevancia social, utilidad teórica y/o práctica en el ámbito de la escuela profesional. Para el grado de doctor se requiere una tesis de máxima rigurosidad académica y de carácter original. Es individual para la obtención de un grado; *es individual o en pares para obtener un título profesional*. Asimismo, en su artículo 25° señala: "El tema debe responder a alguna de las líneas de investigación institucionales de la USS S.A.C.".

Que, acorde a documento de vistos, el Comité de investigación de la Escuela profesional de INGENIERÍA DE SISTEMAS, recomienda aprobar en vías de regularización el tema de Tesis: "ANÁLISIS COMPARATIVO DEL RENDIMIENTO DE CONTROLADORES DE REDES DEFINIDAS POR SOFTWARE DE TIPO OPEN SOURCE PARA LA GESTIÓN DE UNA RED DE COMPUTADORAS" de la línea de investigación de INFRAESTRUCTURA, TECNOLOGÍA Y MEDIO AMBIENTE, a cargo de SANTISTEBAN YNGA BICRY RAUL, en condición de egresado(s), del Programa de estudios de INGENIERÍA DE SISTEMAS, por motivo de deterioro de la resolución de aprobación del tema de tesis N°1061-2019/FIAU-USS.

Estando a lo expuesto, y en uso de las atribuciones conferidas y de conformidad con las normas y reglamentos vigentes;

SE RESUELVE:

ARTÍCULO 1°: APROBAR, en vías de regularización, el tema de Tesis **"ANÁLISIS COMPARATIVO DEL RENDIMIENTO DE CONTROLADORES DE REDES DEFINIDAS POR SOFTWARE DE TIPO OPEN SOURCE PARA LA GESTIÓN DE UNA RED DE COMPUTADORAS",** perteneciente a la línea de investigación de **INFRAESTRUCTURA, TECNOLOGÍA Y MEDIO AMBIENTE**, a cargo de **SANTISTEBAN YNGA BICRY RAUL,** del Programa de estudios de **INGENIERÍA DE SISTEMAS.**

ARTÍCULO 2°: ESTABLECER, que la inscripción del Tema de la Tesis se realice a partir de emitida la presente resolución y tendrá una vigencia de dos (02) años.

ARTÍCULO 3°: DEJAR SIN EFECTO, toda Resolución emitida por la Facultad que se oponga a la presente Resolución.

REGÍSTRESE, COMUNÍQUESE Y ARCHÍVESE

Dr. Mario Fernando Ramos Mesoni
Decano - Facultad de Ingeniería, Arquitectura y Urbanismo
UNIVERSIDAD SEÑOR DE SIPÁN SAC.

MBA. María Noelia Rubio Rivera
Secretaria Académica - Facultad de Ingeniería, Arquitectura y Urbanismo
UNIVERSIDAD SEÑOR DE SIPÁN SAC.

Cc: Interesado, Archivo

Anexo 2. Instrumentos de recolección de datos, con su respectiva validación de los instrumentos.

Tabla 15

Formato del instrumento de recolección de datos

Nombre de Prueba	(Se detalla el nombre de la Prueba)		
Numero de Prueba	(N° Característica)	**Tipo Controlador:**	(Tipo Controlador)
Objetivo/Justificación	(Objetivos de la Prueba		
Secuencia de la Prueba	(Secuencia de la Prueba)		

Tabla 16

Instrumento Ficha de observación

Pruebas/Controlador	Prueba 1	Pruebas Completadas
Controlador 1	1	1
Controlador 2	1	1
Controlador 3	1	1
Controlador 4	1	1
Controlador 5	1	1
		5

Anexo 3. Otros anexos que considere conveniente.

EJECUCIÓN DE TOPOLOGÍA DE RED PERSONALIZADA EN PYTHON

```
"""Topologia red Proyecto SDN ---
---Red Informatica y Telecomunicaciones - Sede Central---
---Core---
---Distribucion Nodos---
---Acceso---
"""
from mininet.topo import Topo
from mininet.net import Mininet
from mininet.util import dumpNodeConnections

class RedUSS( Topo ):
    "Topologia"
    def build( self ):

        # host Red Campus
        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')

        # host Red Telematica
        h11 = self.addHost( 'h11')
        h12 = self.addHost( 'h12')
        h13 = self.addHost( 'h13')

        # host Sede Central
        h10 = self.addHost( 'h10')

        # switch Core
        s1 = self.addSwitch( 's1' )
        s2 = self.addSwitch( 's2' )

        # switch nodos distribucion
        s3 = self.addSwitch( 's3' )
        s4 = self.addSwitch( 's4' )
        s5 = self.addSwitch( 's5' )
        s6 = self.addSwitch( 's6' )
        s7 = self.addSwitch( 's7' )
```

```
s41 = self.addSwitch( 's41' )
s42 = self.addSwitch( 's42' )
s43 = self.addSwitch( 's43' )
s44 = self.addSwitch( 's44' )

# switch acceso
# switch nodo1
s17 = self.addSwitch( 's17' )
s18 = self.addSwitch( 's18' )
s19 = self.addSwitch( 's19' )
s8 = self.addSwitch( 's8' )

# switch nodo2
s24 = self.addSwitch( 's24' )
s25 = self.addSwitch( 's25' )
s26 = self.addSwitch( 's26' )
s9 = self.addSwitch( 's9' )

# switch nodo3
s27 = self.addSwitch( 's27' )
s28 = self.addSwitch( 's28' )
s29 = self.addSwitch( 's29' )
s10 = self.addSwitch( 's10' )

# switch nodo4
s30 = self.addSwitch( 's30' )
s31 = self.addSwitch( 's31' )
s32 = self.addSwitch( 's32' )
s33 = self.addSwitch( 's33' )
s11 = self.addSwitch( 's11' )

# switch nodo5
s12 = self.addSwitch( 's12' )

# switch nodo6
s13 = self.addSwitch( 's13' )

# switch nodo7
s14 = self.addSwitch( 's14' )

# switch nodo8
s34 = self.addSwitch( 's34' )
s35 = self.addSwitch( 's35' )
s36 = self.addSwitch( 's36' )
s37 = self.addSwitch( 's37' )
s15 = self.addSwitch( 's15' )

# switch nodo9
s38 = self.addSwitch( 's38' )
```

```
s39 = self.addSwitch( 's39' )
s40 = self.addSwitch( 's40' )
s16 = self.addSwitch( 's16' )

# switch Red Telematica
s20 = self.addSwitch( 's20' )
s21 = self.addSwitch( 's21' )
s22 = self.addSwitch( 's22' )

# switch sede central
s23 = self.addSwitch( 's23' )

# links entre cores
self.addLink( s1, s2)

# links distribucion a core1
self.addLink( s3, s1)
self.addLink( s4, s1)
self.addLink( s5, s1)
self.addLink( s6, s1)
self.addLink( s7, s1)
self.addLink( s41, s1)

# links distribucion a core2
self.addLink( s42, s2)
self.addLink( s43, s2)
self.addLink( s44, s2)

# link acceso
# link nodo1
self.addLink( s17, s3)
self.addLink( s18, s3)
self.addLink( s19, s3)
self.addLink( s8, s3)

# link nodo2
self.addLink( s24, s4)
self.addLink( s25, s4)
self.addLink( s26, s4)
self.addLink( s9, s4)
# link nodo3
self.addLink( s27, s5)
self.addLink( s28, s5)
self.addLink( s29, s5)
self.addLink( s10, s5)

# link nodo4
self.addLink( s30, s6)
self.addLink( s31, s6)
```

```
self.addLink( s32, s6)
self.addLink( s33, s6)
self.addLink( s11, s6)

# link nodo5
self.addLink( s12, s7)

# link nodo6
self.addLink( s13, s41)

# link nodo7
self.addLink( s14, s42)

# link nodo8
self.addLink( s34, s43)
self.addLink( s35, s43)
self.addLink( s36, s43)
self.addLink( s37, s43)
self.addLink( s15, s43)

# link nodo9
self.addLink( s38, s44)
self.addLink( s39, s44)
self.addLink( s40, s44)
self.addLink( s16, s44)

# link Red Telematica
self.addLink( s20, s1)
self.addLink( s21, s1)
self.addLink( s22, s1)

# link Sede Central
self.addLink( s23, s2)

# link host acceso
self.addLink( h1, s8)
self.addLink( h2, s9)
self.addLink( h3, s10)
self.addLink( h4, s11)
self.addLink( h5, s12)
self.addLink( h6, s13)
self.addLink( h7, s14)
self.addLink( h8, s15)
self.addLink( h9, s16)

# Links host Red Telematica
self.addLink( s20, h11)
self.addLink( s21, h12)
self.addLink( s22, h13)
```

```
        # Links host Sede Central
        self.addLink( s23, h10)

            # Permite al archivo ser importado con `mn --custom
            <filename> --topo minimal`
            topos = {
            'RedUSS': RedUSS
            }
```

INSTALACIÓN DEL CONTROLADOR ONOS

```
sudo adduser sdn --system --group
sudo apt update
sudo apt install git zip curl unzip python-minimal openjdk-8-jdk -y
sudo mkdir -p /opt && cd /opt
sudo wget http://repo1.maven.org/maven2/org/onosproject/onos-
releases/2.0.0/onos-2.0.0.tar.gz
sudo tar xzf onos-2.0.0.tar.gz
sudo mv onos-2.0.0 onos
sudo chown -R sdn:sdn onos
sudo -u sdn nano /opt/onos/options
export ONOS_USER=sdn
export ONOS_APPS=drivers,openflow,gui2

sudo cp /opt/onos/init/onos.initd /etc/init.d/onos
sudo cp /opt/onos/init/onos.service /etc/systemd/system/
sudo systemctl daemon-reload
sudo systemctl enable onos

sudo systemctl start onos

WEB UI : ip_onos:8181/onos/ui/index.html
CLI : ssh -p 8101 onos@ip_onos

Default username : onos
Default password : rocks

/opt/onos/bin/onos-user-password onos --remove
/opt/onos/bin/onos-user-password zufar zufar
generate ssh-keygen dan tambahkan public keynya

ssh-keygen -t rsa
/opt/onos/bin/onos-user-key $USER ~/.ssh/id_rsa.pub
login ke CLI
/opt/onos/bin/onos
```

INSTALACIÓN OPENSSH EN UBUNTU 18.04 Y UBUNTU 16.04

```
sudo apt-get update
sudo apt-get install openssh-server
```

sudo apt-get install openssh-client
sudo systemctl start sshd.service

INSTALACIÓN DE MININET EN UBUNTU 16.04/18.04

git clone https://github.com/mininet/mininet
cd mininet
git tag
git checkout -b 2.2.2
cd ..
mininet/util/install.sh [options]
mininet/util/install.sh –nfv

INSTALACIÓN DEL CONTROLADOR OPENDAYLIGHT

System requirements:
Configuration: 2 Core + 8GB RAM (preferably)
OS:preferencia Linux Ubuntu Version.
Ubuntu versions : Ubuntu 18.04
Java - 8+ version
JAVA Installation
sudo apt-get install default-jdk
ODL Installation
OpenDayLight Version : SODIUM - SR1
wget https://nexus.opendaylight.org/content/repositories/public/org/opendaylight/integration/opendaylight/0.11.1/opendaylight-0.11.1.tar.gz
tar xvzf opendaylight-0.11.1.tar.gz
cd opendaylight-0.11.1/
./bin/karaf

Feature command
Manage the features/plugins

feature:list Lists all existing features available from the defined repositories.
feature:list -i List only installed features
feature:list -i | grep List the installed features and grep with specific name.
feature:list -i | grep restconf
feature:install Installs a feature with the specified name and version.
>feature:install odl-restconf
feature:uninstall
feature:uninstall odl-restconf
feature:info odl-restconf

Log command
log:display
Displays log entries.
log:get
Shows the currently set log level
log:set
Sets the log level.
The log level to set (TRACE, DEBUG, INFO, WARN, ERROR) or DEFAULT to unset
Log file is located in data/log/karaf.log

Exit Command
shutdown, logout

NorthBound Interface - RestConf/Rest AP
NorthBound Interface - RestConf/Rest API
RESTCONF uses HTTP methods to provide CRUD operations on a conceptual datastore containing YANG-defined data, which is compatible with a server that implements NETCONF datastores.

Methods
The RESTCONF protocol uses HTTP methods to identify the CRUD operations requested for a particular resource.
Following are the widely used methods
GET:
The GET method is sent by the client to retrieve data and metadata for a resource. It is supported for all resource types, except operation resources.
If the user is not authorized to read the target resource, an error response containing a "401 Unauthorized" status-line SHOULD be returned.
PUT:
The PUT method is sent by the client to create or replace the target data resource.
A request message-body MUST be present, representing the new data resource.
The "insert" and "point" query parameters MUST be supported by the PUT method for data resources.
if the PUT request creates a new resource, a "201 Created" status-line is returned. If an existing resource is modified, a "204 No Content" status-line is returned.
DELETE:
The DELETE method is used to delete the target resource. If the DELETE request succeeds, a "204 No Content" status-line is returned. If the user is not authorized to delete the target resource, then an error response containing a "403 Forbidden" status-line SHOULD be returned.
POST
The POST method is sent by the client to create a data resource or invoke an operation resource.
3. REST Clients
POSTMAN GUI Based.
To install, follow this link
https://www.postman.com/downloads/
https://linuxize.com/post/how-to-install-postman-on-ubuntu-18-04/
curl Quick to use. Command line interface. Little hard to understand the flags/options
sudo apt-get install curl
4. ODL Setup for demo
Install ODL as per previous session, and start the ODL KARAF shell
./bin/karaf
Install the restconf and Openflow plugins
feature:install odl-restconf
feature:install odl-openflowplugin-flow-services-rest
Default ODL username/password : admin/admin
ODL IP&Port: http://192.168.122.219:8181
GET http://192.168.122.219:8181restconf/operational/network-topology:network-topology/

DataModel: XML

5. Accessing NorthBound RESTCONF/RESTAPI with CURL
GET Method:
curl --user "admin":"admin" -H "Accept: application/xml" -H "Content-Type: application/xml" -X GET http://192.168.122.227:8181/restconf/operational/network-topology:network-topology/

6. Accessing NorthBound RESTCONF/RESTAPI with POSTMAN
http://192.168.122.227:8181/restconf/operational/network-topology:network-topology/
Base Headers:
In the Authorization TAB, Select,
Type: Basic Auth

Fill username and password as mentioned above
and click Preview Request,

In the Header TAB,
you could see "Authorization" Header and value "Basic xxxxxxxxxxxx" could have been populated automatically.

Add "Content-Type" and "Accept" header with Value is "application/xml" for both.

Method: GET and Query

'OPEN SOURCE' APLICADO A REDES

PROYECTOS LIBRES, ABIERTOS Y LISTOS PARA REVOLUCIONAR LAS TELECOMUNICACIONES

Nos encontramos en el medio de una gran transformación para la industria de las telecomunicaciones, no sólo por la llegada de las redes definidas por *software* (SDN) y la virtualización de funciones de red (NFV), sino además porque estas tendencias, tan dependientes de componentes de *software*, han atraído la formación de grandes comunidades de desarrollo abierto a nivel mundial, trayéndonos el "fenómeno Linux" a las redes. La creciente presencia de fabricantes de soluciones de redes tradicionales y grandes proveedores de servicios en dichas comunidades *open source*, incluso desde su concepción, no hace sino confirmar todo el valor detrás de este esquema colaborativo.

¿En qué consiste *Open Source*? aplicado a *software*, es el código fuente que lo conforma, puesto libremente a disposición de todos para modificaciones y mejoras. Entre los ejemplos más populares destacan Linux y Android, sistemas operativos de adopción exitosa y masiva. Hoy el concepto se extiende incluso al *hardware*, a través proyectos como *Open Compute Project*, fundado en el 2011 por Facebook y de creciente relevancia.

¿Cuál es el valor que aporta *Open Source*? la naturaleza abierta de las comunidades que construyen estos productos abiertos, y las licencias que las regulan, permiten que cualquiera pueda aportar mejoras en la medida en que éstas se mantengan a disposición de todos. Se promueve entonces la colaboración e innovación global sin perder de vista los objetivos de la comunidad. Más allá del código, el valor se origina en las personas que conforman la comunidad y dedican esfuerzo a intereses de bien común.

¿Cómo se financian las comunidades *Open Source*? Existen múltiples modelos de financiamiento garantizados por el enorme valor que los diferentes actores de la industria encuentran en las comunidades:

Fabricantes, consultores e integradores	Aceleran su propia innovación, maximizan su productividad y minimizan costos de desarrollo, permitiéndose luego incorporar los componentes abiertos en soluciones propietarias. Desarrollan modelos de soporte, entrenamiento y optimización alrededor de productos *open source*. Todo esto con el objetivo de proveer un valor agregado en el cual sus clientes están dispuestos a invertir.
Universidades, Centros de Investigación y desarrolladores independientes	Quizá los principales originadores de esta tendencia, interesados en aportar al desarrollo científico, y junto con sus comunidades, colaborar abiertamente y construir una buena reputación colectiva e individual en la industria en la que se desenvuelven.
Usuarios finales (empresas e individuos)	Minimizan los costos de inversión y operación haciendo uso libre de productos abiertos, contribuyendo con *feedback* para las comunidades y contratando consultorías y soporte avanzado en caso lo necesiten.

¿Cómo este ecosistema beneficia hoy a las telecomunicaciones? SDN y NFV proponen una nueva forma de construir y operar redes, permitiendo mayor agilidad y flexibilidad para la creación de servicios a través de *software*. *Open Source* potencia estas ventajas, minimizando costos y acelerando la adopción e innovación.

Grandes casos de uso de *Open Source*

AT&T - OpenStack
https://goo.gl/uzjGYh
Uno de los *carriers* con mayor innovación reciente duplicó el uso de software abierto en el último año, y se suma a los miles de exitosos despliegues de OpenStack (https://goo.gl/vqKpCT)

Orange - OpenDaylight
https://goo.gl/ugmO1x
La visión de Orange para el presente año coloca a OpenDaylight como la plataforma de referencia para *Carrier Grade SDN;* decenas de operadores lo confirman (https://goo.gl/62FTvC)

Google - OpenFlow
http://goo.gl/IMJi1F
B4, la red WAN definida por software de Google, un temprano caso de éxito para SDN vía OpenFlow.

Facebook - Open Compute Project
http://goo.gl/v9x9MZ
'Facebook está haciendo por el *hardware* lo que Linux, Android y otros hicieron por el *software*'

Servidores Web públicos - Linux
http://goo.gl/v9x9MZ
El sistema operativo *open source* más importante está presente en el 98% de los servidores web públicos.

DESARROLLANDO TECNOLOGÍAS DE REDES EN FORMA ABIERTA

PRINCIPALES PROYECTOS ABIERTOS EN TELECOMUNICACIONES

APLICACIONES
interfaz para el usuario final, catálogos de servicios y portales de auto-servicio en general.

Cloud Foundry, Cloudify, OpenShift

Entornos de desarrollo para aplicaciones en la nube de tipo PaaS (Plataforma como Servicio). Soportan diversas bases de datos y lenguajes de desarrollo de aplicaciones.

node, AngularJS, Rails

Entornos de desarrollo para aplicaciones web, donde destacan los de JavaScript (Node, Angular, React), Ruby (Rails), Python (Django), PHP (PhalconPHP, Laravel), entre otros.

GESTIÓN DE INFRAESTRUCTURA VIRTUAL
orquestación de redes virtuales

OpenStack (Apache-2.0 / Python)
La aplicación más consolidada y popular para construir entornos *Cloud*, orquestando sistemas virtuales de cómputo, memoria, almacenamiento y red.

Kubernetes (Apache-2.0 / Go) y Mesos (Apache-2.0 / C++)
Automatización del despliegue y operación de clústeres de servidores físicos y contenedores virtuales, ambos integrables con OpenStack.

PLANO DE CONTROL
implementación y control de funciones de red avanzadas

Open Daylight (EPL-1.0 / Java)
El controlador SDN con mayor amplitud y soporte, respaldado por una comunidad con la misión de liderar la transformación hacia el despliegue de SDN abierto.

ONOS (Apache-2.0 / Java)
Un controlador SDN emergente, muy cercano a ODL en popularidad, principalmente enfocado en redes de ISP y en la visión de CORD (transformación de la red hacia una arquitectura de Centro de Datos)

PROTOCOLO DE CONTROL
mecanismo de comunicación entre controladores y planos de datos

OpenFlow, el primer estándar SDN que hizo realidad la independización de los planos de control y datos. El protocolo es ampliamente soportado y permite la definición y configuración de funciones de red avanzadas.

Netconf es un protocolo de configuración para equipos de red. En el 2010 surgió un lenguaje de modelos de datos para este protocolo, llamado **Yang**, al que cada vez más fabricantes se están adhiriendo para uniformizar su sintaxis.

OVSDB es el protocolo de configuración para switches Open vSwitch.

PLANO DE DATOS
sistemas de conmutación de paquetes

DPDK (BSD-3-Clause / C)
Es un conjunto de librerías para lograr un rápido procesamiento de paquetes sobre cualquier procesador.

FD.io (Apache-2.0 / C)
Utiliza DPDK, VPP y otras tecnologías para lograr la mayor velocidad y menor latencia sobre cualquier procesador.

Open vSwitch (Apache2.0 / C)
se trata de un switch virtual multi-capa para entornos en producción y diversos tipos de interfaces programáticas.

CAPA FÍSICA
hardware que soporta la conmutación de paquetes

Open Compute Project, la comunidad que intercambia especificaciones de diseño para dispositivos de red abiertos y programables así como sus accesorios, incluyendo servidores, unidades de almacenamiento, bastidores, fuentes de energía y switches.

***Whiteboxes*: switches y servidores de uso genérico**
Con base en las especificaciones del Open Compute Project, lineamientos OpenFlow u otros, diversos fabricantes desarrollan switches y servidores con interfaces abiertas y programáticas, de manera que el usuario implemente en ellos las funciones que requiera a través de controladores SDN.

HIPERVISORES

KVM, Xen

KVM y **XEN** son los hipervisores abiertos más populares para crear máquinas virtuales sin depender de VMWare o Microsoft.

CONTENEDORES

docker, CoreOS

Docker y **CoreOS**, los proyectos principales para la gestión de contenedores Linux virtuales, gobernados por el Open Container Initiative.

SISTEMA OPERATIVO

Linux, el sistema operativo abierto por excelencia, consolidado como favorito para el soporte de aplicaciones de red en todas las capas, siendo Ubuntu y Debian las distribuciones más utilizadas (~41% según linuxcounter)

Table 1: Feature-bade comparison of SDN controllers

	Ryu	Floodlight	OpenDayLight	ONOS
Southbound Interfaces	OF 1.0, 1.2, 1.3, 1.4, NETCONF, OFCONFIG, OVSDB	OF1.0,1.1,1.2, 1.3, 1.4,1.5	OF1.0, 1.3,1.4,1.5 NETCONF/YANG,OVSDB, PCEP, BGP/LS, LISP, SNMP, OFCONFIG	OF1.0, 1.3,1.4, 1.5 NETCONF
REST API	Yes (For SB only)	Yes	Yes	Yes
GUI	Yes (Initial phase)	Web/ Java- based	Web-based	Web-based
Modularity	Medium	Medium	High	High
Orchestrator Support	Yes	Yes	Yes	No
OS Support	Most supported on Linux	Linux, Windows , and MAC	Linux, Windows , and MAC	Linux, Windows , and MAC
Partner	Nippo Telegraph And Telephone Corporation (NTT)	Big Switch Networks	Linux Foundation With Memberships Covering Over 40 Companies, like Cisco, IBM,	ON.LAB, Sk Telecom, Cisco, Ericsson, Fujitsu, Huawei, Intel, AT&T, Nec, Ciena, Nsf.Ntt Comunnication
Documentation	Medium	Good	Very good	Good
Programming Language	Python	Java	Java	Java
Multi-threading support	Yes	Yes	Yes	Yes
TLS Support	Yes	Yes	Yes	Yes
Virtualization	Mininet and OVS	Mininet and OVS	Mininet and OVS	Mininet and OVS
Application Domain	Campus	Campus	Data center and Transport-SDN WAN	Data center and Transport-SDN WAN
Distributed/Centralized	Centralized	Centralized	Distributed	Distributed

TABLE II
CONTROLLER PROGRAMMING LANGUAGES AND DOMAIN OF APPLICATION

Language	Controller	Domain	Thread
Java	OpenDayLight	Data Center	Multi
Java	ONOS	Telecom / Data Center	Multi
Java	Floodlight	Campus	Multi
Java	Iris	Telecom	Multi
Python	Ryu	Campus / Research	Single
C	MUL	Data Center	Multi
C++	Runos	Telecom / Data Center	Multi
C, Python	ONIX	WAN / Data Center	Multi
C, Ruby	Trema	Data Center	Multi

Table I: Qualitative Comparison

	POX	RYU	ODL	ONOS
NB interface	Yes (Limited)	Yes	Yes	Yes
SB interface	OF1.0	OF & Other SB Protocols	OF & Other SB Protocols	OF & Other SB Protocols
GUI	No	Yes (Limited)	Yes (Web Based)	Yes (Web Based)
REST API	Yes	Yes	Yes	Yes
Programming Language	Python	Python	Java	Java
OpenFlow Support	OF v1.0	OF v1.0, 1.2, 1.3, 1.4, 1.5	OF v1.0, 1.3,1.4	OF v1.0, 1.3
Modularity	Low	Low	High	High
Multi-threading support	No	No	Yes	Yes
Documentation	Low	Low	High	High

TABLE 1. CONTROLLERS' ATTRIBUTE COMPARISON

Attributes	Beacon	OpenDaylight	ONOS
Objective	Performance, Ease of applications development, Runtime configuration	Modular, Pluggable, Flexible controller	SDN OS for network providers. Modular, Scalability, High availability, Performance,
Architecture	Centralized OGSI-based	Distributed, Cluster-based	Distributed, Three tiers
Core Network Services	Core network services	Controller platform	Core services and Distributed core layer
Southbound API/protocols	OpenFlowj, Device manager	OpenFlow, ovsdb, netconf, pcep,snmp,bgp	OpenFlow, NETCONF, PCEP
Northbound API	Ad-hoc API	REST API	Application Intents
East/Westbound API	Not yet provided	Not yet provided	Not yet provided
Service Abstraction Layer / Plug-ins	OSGI plug-ins	Support additional protocols	Southbound API
Management Interfaces	Web UI	Java	Java
Programming Language	Java	Java	Java

Table 1: Feature based comparison of the most popular open source SDN controllers

	Programming Language	GUI	Documentation	Modularity	Distributed/ Centralized	Platform Support	Southbound APIs	Northbound APIs	Partner	Multithreading Support	OpenStack Support	Application Domain
ONOS	Java	Web Based	Good	High	D	Linux, MAC OS, And Windows	OF1.0, 1.3, NETCONF	REST API	ON.LAB, AT&T, Ciena, Cisco, Ericsson, Fujitsu, Huawei, Intel, Nec, NsfNet Communication, Sk Telecom	Y	N	Datacenter, WAN and Transport
Open-Day-Light	Java	Web Based	Very Good	High	D	Linux, MAC OS, And Windows	OF1.0, 1.3, 1.4, NET-CONF/ YANG, OVSDB, PCEP, BGP/LS, LISP, SNMP	REST API	Linux Foundation With Memberships Covering Over 40 Companies, Such As Cisco, IBM, NEC	Y	Y	Datacenter
NOX	C++	Python + QT4	Poor	Low	C	Most Supported On Linux	OF 1.0	REST API	Nicira	NOX _MT	N	Campus
POX	Pyth on	Python + QT4	Poor	Low	C	Linux, MAC OS, And Windows	OF 1.0	REST API	Nicira	N	N	Campus
RYU	Pyth on	Yes	Fair	Fair	C	Most Supported On Linux	OF 1.0, 1.2, 1.3, 1.4, NETCONF, OF-CONFIG	REST For Southb ound	Nippo Telegraph And Telephone Corporation	Y	Y	Campus
Beacon	Java	Web Based	Fair	Fair	C	Linux, MAC OS, And Windows	OF 1.0	REST API	Stanford University	Y	N	Research
Maestro	Java	-	Poor	Fair	C	Linux, MAC OS, And Windows	OF 1.0	REST API	RICE, NSF	Y	N	Research
Flood-Light	Java	Web/ Java Based	Good	Fair	C	Linux, MAC OS, And Windows	OF 1.0 , 1.3	REST API	Big Switch Networks	Y	N	Campus
Iris	Java	Web Based	Fair	Fair	C	Linux, MAC OS, And Windows	OF 1.0, 1.3, OVSDB	REST API	ETRI	Y	N	Carrier-Grade
MUL	C	Web Based	Fair	Fair	C	Most Supported On Linux	OF 1.4, 1.3, 1.0, OVSDB, OF-CONFIG	REST API	Kulcloud	Y	Y	Datacenter
Ruuo	C++	Web Based	Fair	Fair	D	Most Supported On Linux	OF 1.3	REST API	ARCCN	Y	N	WAN, Telecom and Datacenter
Lib-Fluid	C++	-	Fair	Fair	-	Most Supported On Linux	OF 1.0, 1.3	-	ONF	Y	N	-

TABELA I. ANÁLISE COMPARATIVA DOS CONTROLADORES

Controlador	Parâmetros Comparativos						
	Ano	*Linguagem de Programação*	*Plataforma*	*Interface*	*Documentação*	*Versão OF*	*Exemplos de Áreas de Trabalhos Desenvolvidos*
Beacon	2010	Java	Linux, MAC OS e Windows	Web	Razoável	OpenFlow 1.0	ET [4], M [5], W [6] e S [7]
IRIS	2014	Java	Linux, MAC OS e Windows	Web e Java	Razoável	OpenFlow 1.0.1 até 1.3.2	RDC [8]
NOX	2008	C++	Linux	PyQT4	Fraco	OpenFlow 1.0	ET [9], M [10], RDC [11] e S [12]
POX	2012	Python	Linux, MAC OS e Windows	PyQT4	Fraco	OpenFlow 1.0, parcialmente 1.1	ET [13], M [14] e S [15]
Floodlight	2012	Java	Linux, MAC OS e Windows	Web e Java	Bom	OpenFlow 1.0 até 1.5	ET [16], W [17], M [18], RDC [19] e S [20]
OpenDayLight	2013	Java	Linux, MAC OS e Windows	Web	Bom	OpenFlow 1.0 até 1.4	ET [21], M [22], RDC [23]
ONOS	2015	Java	Linux, MAC OS e Windows	Web	Bom	OpenFlow 1.0 até 1.5	M [24]
Ryu	2013	Python	Linux	GUI Patch	Razoável	OpenFlow 1.0 até 1.5	ET [25]
OpenMUL	2012	C	Linux	Web	Razoável	OpenFlow 1.4	RDC [26]
Trema	2011	C/C++ e Ruby	Linux	N/A	Fraco	OpenFlow 1.0 até 1.3	ET [27]

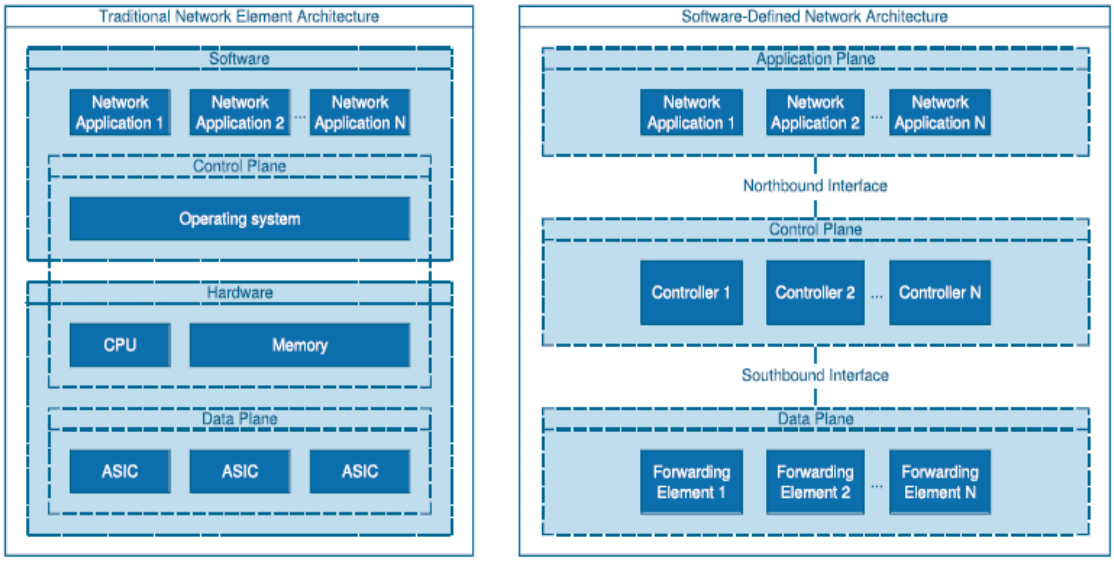

FIGURE 1. Migration from traditional to software-defined network architecture.

Introducción: Estado de las redes para la era digital

Resumen de la sección

Aportes clave

- Las tendencias como la globalización, la transformación digital, la automatización y la resiliencia del negocio, además de la sustentabilidad, están dando forma a los requisitos necesarios para lograr un nuevo tipo de red.
- El cambiante panorama tecnológico: modelos emergentes nativos de la nube, Internet de las Cosas, inteligencia artificial (IA), dispositivos móviles, amenazas de ciberseguridad y aplicaciones inmersivas, está afectando drásticamente las arquitecturas y operaciones de redes de TI.
- La enorme escala, complejidad y naturaleza dinámica de estas demandas superan la capacidad de los operadores humanos por sí solos.
- Las nuevas redes utilizan tecnologías emergentes como la IA (Inteligencia Artificial), el aprendizaje automático y la automatización para simplificar y proteger las operaciones, permitir una rápida adaptabilidad y aumentar la toma de decisiones de las personas.

Tendencias empresariales y tecnológicas globales que están dando forma a la nueva red

700M	50%	14.6B	42%	53%	12X
de contenedores alojados en el borde para el año 2021[1]	de las cargas de trabajo fuera del centro de datos empresarial para el 2021[2]	de dispositivos de IoT para el año 2022[3]	de crecimiento anual del tráfico móvil empresarial, de 2017 a 2022[3]	de ataques de ciberseguridad ocasionan más de US$500,000 en daños[4]	de aumento en el tráfico de AR/VR (Realidad Aumentada/ Realidad Virtual) para 2022[3]