compAS
Grupo de capacitación e investigación pedagógica

ANÁLISIS DE LAS REDES DEFINIDAS POR SOFTWARE (SDN) FRENTE A REDES TCP/IP Y COMBINADAS

Zhuma Mera, Emilio
Guzmán Vélez, Dayana Mariuxi
Cáceres Miranda, Carlos Andrés
Oviedo Bayas, Byron Wladimir

ANÁLISIS DE LAS REDES DEFINIDAS POR SOFTWARE (SDN) FRENTE A REDES TCP/IP Y COMBINADAS

Zhuma Mera, Emilio
Guzmán Vélez, Dayana Mariuxi
Cáceres Miranda, Carlos Andrés
Oviedo Bayas, Byron Wladimir

ANÁLISIS DE LAS REDES DEFINIDAS POR SOFTWARE (SDN) FRENTE A REDES TCP/IP Y COMBINADAS

compÁs
Grupo de capacitación e investigación pedagógica

Título original: ANÁLISIS DE LAS REDES DEFINIDAS
POR SOFTWARE (SDN) FRENTE A
REDES TCP/IP Y COMBINADAS

Primera edición: marzo 2020

Zhuma Mera, Emilio
Guzmán Vélez, Dayana Mariuxi
Cáceres Miranda, Carlos Andrés
Oviedo Bayas, Byron Wladimir

Publicado por acuerdo con los autores.

Segundo Congreso Internacional de Sociedad y Tecnología
de la información en la Educación Superior
Guayaquil-Ecuador

Editado en Guayaquil - Ecuador

ISBN: 978-9942-33-199-1

compÁs
Grupo de capacitación e investigación pedagógica

Cita.
E. Zhuma, D. Guzmán, C. Cáceres, B. Oviedo (2020) ANÁLISIS DE LAS REDES DEFINIDAS POR SOFTWARE (SDN) FRENTE A REDES TCP/IP Y COMBINADAS, Editorial Grupo Compás, Universidad Técnica Estatal de Quevedo. Guayaquil Ecuador, 129 pag

Índice

Prólogo 3
Introducción 5
Capítulo 1 10
Situación y realidad 10
Capítulo 2 14
Conceptualizaciones y teorías 14
Capítulo 3 81
Identificación de protocolos de redes SDN. 81
Identificación de Controladores de redes SDN. 82
Identificación de Emuladores. 84
Diseño e implementación de escenarios prácticos para evaluar el desempeño de redes SDN frente a las redes basadas en arquitectura TCP/IP. 88
Escenario Dos con arquitectura TCP/IP. 100
Comparación estadística de resultados del escenario Dos... 102
Tercer Escenario. 105
Cuarto Escenario. 109
Etapa tres: Análisis de tráfico del Tercer Escenario. 116
Determinación de umbral de transmisión óptima. 116
Protocolos involucrados durante la transmisión. 117
Análisis de captura de tráfico con Wireshark. 118
BIBLIOGRAFÍA 129

Prólogo

El libro está enfocado al análisis del desempeño de arquitecturas basadas en redes definidas por software y en arquitectura TCP/IP. Las redes definidas por software (SDN) son un conjunto de redes altamente eficiente, escalable, programable, con gran velocidad y capacidad de gestionar recursos de red. En este proyecto primero se plantea identificar y seleccionar protocolos, controladores y herramientas de emulación que se utilicen actualmente en redes SDN. Se propone diseñar e implementar escenarios para pruebas basados en métricas de evaluación.

En total, se implementan cuarto escenarios utilizando los emuladores Mininet y GNS3, en el primero se mide cual es el ancho de banda que puede soportar cada red utilizando un el software Jperf. El segundo escenario consiste en realizar pruebas de envío de paquetes para medir latencia mínima, media y máxima que puede tener la red, además también se mide el Jitter o variación de paquetes. En el tercer escenario se realiza la transmisión de un streaming de video entre dos hosts utilizando un servidor, este escenario se implementó en GNS3. El cuarto escenario es similar que el escenario dos, se hace el mismo procedimiento para medir latencia y Jitter pero aquí se compara una topología híbrida, SDN y TCP/IP. Cabe destacar que el código para realizar las comparaciones del escenario dos y cuatro se realizó en el software RStudio, que utiliza lenguaje de programación R y es un potente analizador estadístico.

Finalmente se concluye este proyecto de investigación con los resultados, se pudo elegir las herramientas que mejor se adaptaron al proyecto, en este caso el controlador

OpenDayLight, el protocolo OpenFlow y los emuladores Mininet y GNS3. En el primer escenario SDN obtuvo casi el doble de capacidad que TCP/IP para manejar el tráfico UDP. En el segundo escenario las latencias y Jitter resultaron ser estadísticamente iguales en SDN y TCP/IP. El tercer escenario dio resultados favorables para SDN, debido a que la cantidad de datos transmitidos se mantuvo por encima del umbral de transmisión, provocando que el video no tenga pérdidas de calidad en comparación con TCP/IP donde si hubo pérdidas. En el cuarto escenario la topología que se desempeñó mejor en cuanto a latencias y Jitter fue SDN, seguida de la híbrida y TCP/IP.

Introducción

Las telecomunicaciones es un campo que evoluciona constantemente, lo que ha llevado a la aparición de nuevos estándares y arquitecturas dentro de redes LAN[1], MAN[2] y WAN[3], todo esto con la finalidad de lograr mantenernos conectados todo el tiempo y poder utilizar infinidad de servicios. Esta evolución ha ido de la mano con el desarrollo de internet en cada una de sus etapas, actualmente estamos entrando a la llamada última fase de su evolución denominada el "Internet del todo" que trata de enfocarse en conectar casi cualquier cosa a internet y formar una red más centralizada.

La arquitectura de las redes TCP/IP[4] no logra satisfacer las necesidades de los usuarios debido a la poca escalabilidad que poseen y al surgimiento de nuevos servicios como televisión IP (IPTV), el streaming de audio y video, transmisiones de video en ultra alta definición, virtualización de equipos, cloud computing, e incluso el internet de las cosas (IoT). Todos estos servicios corriendo bajo las arquitecturas de redes actuales conllevan a un problema grande debido a que no están optimizadas para manejar las altas velocidades que requieren todos estos servicios.

En los últimos años las redes definidas por software o conocidas por sus siglas en inglés como SDN[5], han creado un nuevo paradigma en redes de telecomunicaciones que cumple con los requerimientos de las redes actuales. Este tipo

[1] Red de área local
[2] Red de área metropolitana
[3] Red de área amplia
[4] Protocolo de control de transporte /protocolo de internet
[5] Red definida por software

de redes, brindan grandes beneficios como tener una red altamente eficiente, escalable, programable, capaz de establecer y gestionar los recursos, es de bajo costo y con gran velocidad [1].

En la actualidad se necesitan redes que sean fáciles de manejar y que vayan cambiando acorde a las necesidades de los usuarios [1]. Las redes definidas por software tienen un gran impacto en los centros de datos debido a la capacidad de virtualización que tienen, esto implica mayor seguridad de la información en red y ahorro en costos de equipos.

Según Jiménez [2], en su trabajo denominado "Redes de distribución de contenidos basadas en redes definidas por software" realizado en Europa, la aplicación de contenidos en redes definidas por software empleando el concepto de balanceo de carga trae consigo una gran reducción del tráfico en la red.

El trabajo acerca de la evolución y contribución para el Internet de las Cosas por las redes definidas por software [3] propuesto por González, Flauzac y Nolot investigadores de Panamá y Francia, consiste en los beneficios de redes definidas por software para IoT como son la centralización, flexibilidad, escalabilidad, abstracción de equipos y programabilidad. Además, diseñan una plataforma SDN-IoT para evaluar escenarios experimentales.

Un grupo de investigadores de la Universidad Tecnológica Nacional de Argentina realizaron un artículo relacionado a la evaluación de atributos de redes definidas por software identificando tráfico [4]. En este artículo se analiza la arquitectura de SDN y se evalúan datos tanto cualitativos como cuantitativos mediante un sistema de decisiones y

simulaciones de tráfico. Esto dio como resultado una red con la capacidad de tomar decisiones para prevenir problemas de recursos de networking.

En un estudio del desempeño de redes definidas por software en entornos LAN [5] en Cuba, se realizaron experimentos con redes de distintos tamaños para medir su comportamiento con diferentes factores. Esto demostró que las redes SDN son escalables y con gran adaptación en el crecimiento de la red.

En el Ecuador se han realizado estudios sobre redes definidas por software, algunos de ellos se detallan a continuación: En el trabajo de Albán y Brito[6], se implementa una red SDN con el uso del emulador Mininet y un controlador para gestionar la red de manera centralizada para asegurar la optimización de recursos.

El trabajo de investigación de Escobar [7], consiste en el control de flujo de una red definida por software usando sensores térmicos. Se controló una red a distancia tomando en cuenta variables externas como la temperatura, lo cual permitió brindar autonomía a la red para evitar que equipos tengan alto grado de temperatura y fallos en su comportamiento.

Siguiendo la misma línea de trabajos realizados en Ecuador, se encuentra el de Cordero[8], en el cual se implementó una red SDN en un entorno virtual aplicando protocolos IPv4 e IPv6[6] con el fin de tener mayor seguridad, reduciendo ataques y mejorar el control y monitoreo de la red.

[6] Protocolo de internet versión 4 y versión 6

El presente proyecto presenta una propuesta de análisis del desempeño de redes definidas por software frente a redes con arquitectura TCP/IP mediante escenarios prácticos utilizando diferentes métricas de evaluación para verificar calidad de servicio de estas redes.

Capítulo 1

Situación y realidad

El Internet no es más que la interconexión de redes más grande a nivel global bajo una arquitectura que ha venido funcionando desde hace unas décadas atrás. Esta arquitectura está basada en el modelo de referencia OSI (Interconexión de sistemas abiertos) creado en 1980 por la Organización internacional de normalización (ISO) [9]. Cuando se comenzaron a desarrollar las primeras redes estas funcionaban correctamente debido a que el tráfico generado en la época era poco y las redes no estaban tan exigidas en su funcionamiento. Pero a medida que Internet se fue popularizando y prácticamente se convertía en una necesidad para las organizaciones y demás usuarios finales, el tráfico de red iba en aumento debido a los dispositivos que se iban conectado a estas redes.

En la actualidad Internet ha pasado de ser una tecnología de uso exclusivo a una necesidad debido a que la mayoría de los dispositivos desarrollados necesitan internet para su funcionamiento. Este cambio en el uso del internet ha sido uno de los problemas más latentes en las redes debido a la aparición cifras de tráfico en la red bastante abrumadoras.

Las empresas proveedoras de internet en su medida tratan de abordar el problema invirtiendo en más infraestructura y equipos modernos, pero esto es una solución momentánea que no resuelve el problema y además es muy costosa. Adicionalmente una red más grande se vuelve compleja lo que hace más difícil su administración y monitoreo. Otros de los problemas que tienen los administradores de red son los cambios de políticas de administración cuando una red está

formada debido a que la configuración debe realizarse en los equipos manualmente y puede albergar errores por parte del administrador.

En un mundo enfocado a la conexión de gran cantidad de dispositivos dentro un mismo entorno, las redes basadas en arquitecturas TCP/IP son las que están presentes tanto en redes mundiales como en redes locales. Pero, a pesar de ser muy utilizadas están teniendo problemas en las redes de nueva generación debido a que fueron pensadas para una sociedad de 40 años atrás. Este crecimiento de las necesidades de los usuarios a nivel global ha empujado a los fabricantes a seguir innovando para adaptarse a los usuarios. El inconveniente de esta innovación es que cada vez es más difícil de cubrir los requerimientos y las nuevas soluciones requieren hardware de mayor costo. Entre otras problemáticas está la seguridad de los datos, la integración de políticas y la administración.

Ante todos estos problemas se están desarrollando soluciones cambiando en su totalidad el enfoque de las redes desde su arquitectura, entre proyectos prometedores está la virtualización del hardware de conectividad y la administración de los mismos a través de software. Este enfoque se lo lleva acabo con las redes SDN, una propuesta que promete acaparar la mayoría de problemas de la actualidad. SDN no es un concepto nuevo, pero en estos últimos años ha recibido un gran apoyo y desarrollo lo cual se ve plasmado en algunas de sus soluciones para las redes WAN principalmente.

El problema principal con SDN es que aún está en desarrollo y por parte de los principales proveedores no recomiendan aun su implementación en redes empresariales, esto se debe

a que no existe un consenso para delimitar hasta donde llegar cuando es aceptable su uso y cuando no.

En un informe denominado "Cisco Visual Networking Index: Forecast and Trends" realizado y publicado en el 2018 por Cisco el tráfico de internet desde el 2017 al 2022 aumentará en un 26% a nivel mundial, el tráfico generado por habitante en promedio aumentará de 16,2GB a 49,8GB mensualmente, así mismo las velocidades de transferencia en promedio subiría de 39Mbps a 75,4Mbps [10].

Por ello la idea de implementar redes centralizadas, de fácil administración y manejo de tráfico por medio de políticas se ha convertido en algo indispensable en estos últimos años.

En el análisis del desempeño de redes definidas por software y de redes con arquitectura TCP/IP se pronostica tener una respuesta favorable en la evaluación de diferentes métricas como latencia, jitter y ancho de banda en ambas redes; pero con un mejor desempeño para el caso de las redes SDN.

Las redes definidas por software son utilizadas para la creación de redes altamente disponibles y gestionadas por un controlador. Estas redes no presentan las limitaciones de las redes actuales como la escalabilidad, la complejidad de definir reglas de seguridad, y la administración del ancho de banda, brindándole al administrador de la red la facilidad de gestionar diferentes tipos de servicios.

Las redes como están estructuradas hoy en día no están preparadas para soportar estos volúmenes de tráfico debido a su incapacidad de escalabilidad. Existen grandes cantidades de tráfico generado en las redes por el uso de dispositivos que se conectan a Internet. Se estima que en un par de años el tráfico va a aumentar en un 26% a nivel

mundial causando un gran inconveniente en la capacidad que tienen las redes actualmente [10]. Por ello se está optando por las redes definidas por software que tienen más capacidad de administración y gestión.

Este tipo de redes tienen la ventaja de presentar mayor seguridad debido a que todos los equipos de la red se configuran mediante el controlador y no se tiene la necesidad de configurarlos individualmente. Otra ventaja que presentan es la capacidad del manejo del tráfico, en dónde se ven beneficiadas empresas que trabajan con grandes cantidades de información como son los proveedores de servicios de telecomunicaciones, data centers, entre otras [11]. Las redes definidas por software son las que trabajarán en un futuro con las redes 5G por ser escalables, altamente eficientes y cumplir con todas los requerimientos de alta velocidad [12]. Utilizar SDN da una perspectiva diferente cuando creamos redes, debido a que solo se utiliza software para controlar los diferentes recursos de red. Esto es una solución viable para el problema de costos por implementación de equipos y brinda la posibilidad de implementar equipos de diferentes fabricantes sin inconvenientes de compatibilidad de estándares y protocolos [13]. El presente proyecto de investigación busca brindar una solución a inconvenientes de redes basadas en arquitectura TCP/IP, haciendo un análisis del desempeño de SDN utilizando escenarios prácticos realizados con la herramienta de emulación Mininet. Además, se implementará una red SDN con equipos que utilicen el protocolo Openflow, en donde se medirán parámetros en la transferencia de datos en red.

Capítulo 2

Conceptualizaciones y teorías

Redes.

Las redes en telecomunicaciones se las define como la interconexión de varios dispositivos para realizar intercambio de datos a través de señales electromagnéticas u ópticas ya sea a corta o larga distancia. Por lo general las redes están basadas en infraestructuras de trabajo por medio de cables o mediante el uso de espectro de radio frecuencia [14].

Topología de red.

Es un arreglo lógico o físico de dispositivos que van interconectados entre sí a través de uno a varios medios de comunicación con el propósito de intercambiar información. Existen distintos tipos de topología caracterizados por su simplicidad, capacidad y costo.

Arquitectura de red.

Se define como arquitectura de red a la infraestructura, servicios y protocolos que trabajan bajo un mismo entorno con el objetivo de enviar información a todos los dispositivos, entre las características que debe tener una arquitectura de red en la actualidad para salvaguardar las necesidades de los usuarios están:

- Tolerancia a fallos.
- Escalabilidad.
- Calidad de Servicio (QoS).
- Seguridad [16].

Protocolo.

Es el conjunto de normativas y criterios que especifican como deben comunicarse los diversos componentes dentro una arquitectura de red para el intercambio de datos. Gracias a los protocolos los dispositivos logran definir parámetros en función de los requerimientos de cada dispositivo y sistemas. También establece mecanismos de recuperación de información en caso de pérdida de datos [15].

Virtualización.

La virtualización es una tecnología que permite crear diversos escenarios con recursos dedicados a partir de un solo sistema de hardware. Todo esto lo hace a través de un monitor de máquina virtual el cual permite conectarse directamente con el hardware del host anfitrión para lograr crear entornos virtuales separados del sistema anfitrión en función de la capacidad y requerimientos. Entre los recursos que se pueden compartir del host anfitrión están la CPU[7], Adaptadores de red, Almacenamiento, entre otros [17]. Existen diferentes tipos de virtualizaciones las cuales son:

- **Virtualización de datos:** Permite a las empresas tratar los datos como un vínculo de abastecimiento, con el objetivo de reunir datos de varias fuentes integrarlas entre sí dependiendo de los requerimientos de los usuarios.
- **Virtualización de escritorios:** Permite que un administrador de redes implemente varios ambientes simulados de escritorio en centenas de ordenadores a la vez.

[7] Unidad central de procesamiento

- **Virtualización de servidores:** Virtualizar un servidor facilita la implementación de determinadas funciones bajo un mismo hardware y poderlas ejecutar al mismo tiempo.
- **Virtualización de sistemas operativos:** Se trabaja en el Kernel[8] del sistema anfitrión, con la intención de que un administrador pueda ejecutar entornos Linux y Windows de manera paralela. Esto da beneficios como el costo de implementación, seguridad y reducción de tiempo destinado a servicios de SO.
- **Virtualización de funciones de red:** Separa ciertas funciones de la red para poder distribuirlas en entornos virtuales. Esto ayuda a reducir principalmente la cantidad de componentes físicos como router, cables, switch entre otros [17].

Emulación.

Un emulador es un software informático que tiene como objetivo crear un escenario en el cual recrea el funcionamiento de un equipo o dispositivo como si este estuviera físicamente, igual usa recursos físicos del anfitrión.

Simulación.

La simulación es una técnica que recrea el funcionamiento de un programa dentro de otro sistema que no es compatible o es de otra tecnología. Este término está bastante ligado a la emulación, pero este solo se encarga de software.

[8] Núcleo de un sistema

Modelos de referencia.

En este punto se analiza los modelos de referencia más importantes en las redes de computadoras, el modelo de referencia OSI y el modelo de referencia TCP/IP. Este modelo de capas fue planteado en el modelo de referencia OSI, el cual contaba con 7 capas. Luego surgió el desarrollo de la arquitectura TCP/IP que reagrupaba las capas de modelo de referencia OSI obteniendo una reducción a 4 capas y ese modelo es el que se ha mantenido hasta las redes actuales [15].

Modelo de referencia OSI.

Este modelo está basado en una propuesta planteada por la Organización Internacional de Normas (ISO) como iniciativa para llegar a la estandarización internacional, debido a la acepción de la propuesta de modelos de capas [15]. Este modelo consta de siete capas, las cuales albergan un conjunto de protocolos que trabajan colectivamente para llevar los paquetes de datos desde un punto a otro..

Modelo de referencia TCP/IP.

El modelo de referencia TCP/IP es utilizado desde hace muchos años en las redes de área amplia, en la actualidad la arquitectura de la mayoría de redes está basada en este modelo de referencia debido a su estructura. Este modelo de referencia a diferencia de la anterior cuenta solo con cuatro capas las cuales están distribuidas de la siguiente forma:

Redes definidas por software (SDN).

Las redes definidas por software (SDN), es un concepto de arquitectura de red diferente que separa el plano de control

y el plano de datos para obtener una red enfocada a solucionar los problemas existentes en las redes tradicionales. Este modelo es altamente escalable debido que permite y admite muchas tecnologías actuales independientemente de los fabricantes y está pensado para lograr una red más administrable y sencilla de entender, también es de bajo costo debido a que el enfoque principal es la virtualización de hardware en la nube [18].

Las redes SDN están pensadas en ser programables y autómatas con el fin de lograr eficiencia y garantizar calidad de servicio (QoS), gracias a que la red está centralizada y toda la administración se realiza desde un único controlador [18].

Beneficios de las redes definidas por software (SDN).

Las redes SDN están pensadas para abordar muchas temáticas de las cuales las redes tradicionales carecen, a continuación, se presentarán algunas:

- **Escalabilidad.**

  SDN al ser un modelo dinámico se convierten en redes altamente escalables, debido a que no necesitan realizar cambios excesivos al conectar un nuevo dispositivo o equipo para que la red funcione sin problemas [1].

- **Estructuras menos complejas.**

  El enfoque de virtualización que tiene lleva a las organizaciones a no depender directamente del hardware que tengan porque todos sus equipos estarán alojados en la nube [18].

- **Redes Programables.**

  La administración y gestión de la red se vería favorecida debido a que al no depender de un fabricante, los procesos de administración se los pueden llevar de manera más personalizable permitiendo incluso desarrollar software de administración adaptado a cada necesidad [18].

- **Redes centralizadas.**

  Estas redes trabajan bajo un único controlador que permite la concentración y la gestión de la red en un solo punto [1].

  Arquitectura de redes definidas por software.

  La arquitectura de las redes definidas por software contiene tres capas (Ilustración 1) las cuales cumplen funciones específicas tal como lo establece la ONF[9], fundación sin fines de lucro que se encarga de impulsar el desarrollo de las SDN. Estas etapas se detallarán a continuación:

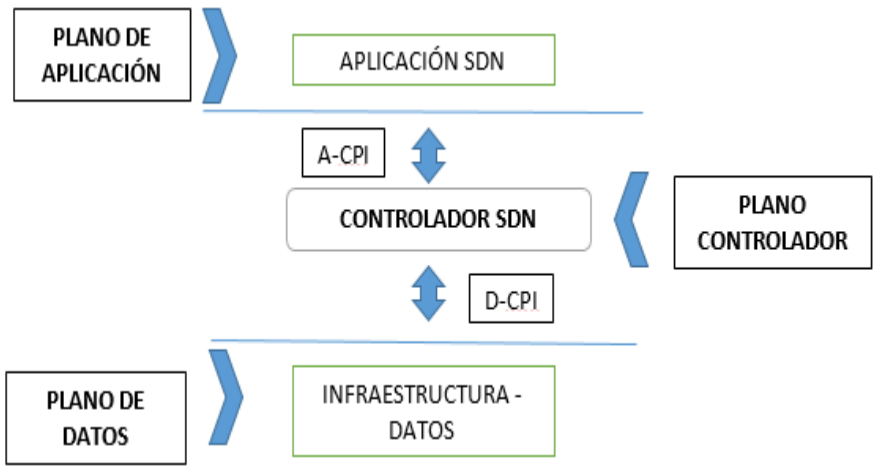


Ilustración 1 Arquitectura SDN. Fuente: [19]

- **Plano de Datos o arquitectura.**

  Es la capa se encarga de incorporar los recursos que interactúan con los clientes, dentro de los recursos que esta concentra están la virtualización, conectividad, seguridad,

[9] Open Networking foundation: fundación sin fines de lucro enfocada al desarrollo de SDN

disponibilidad y calidad de servicio (ilustración 2). Esta capa cumple la función de enviar datos hacia la capa de controlador y así mismo revivir datos para los clientes y recursos [20].

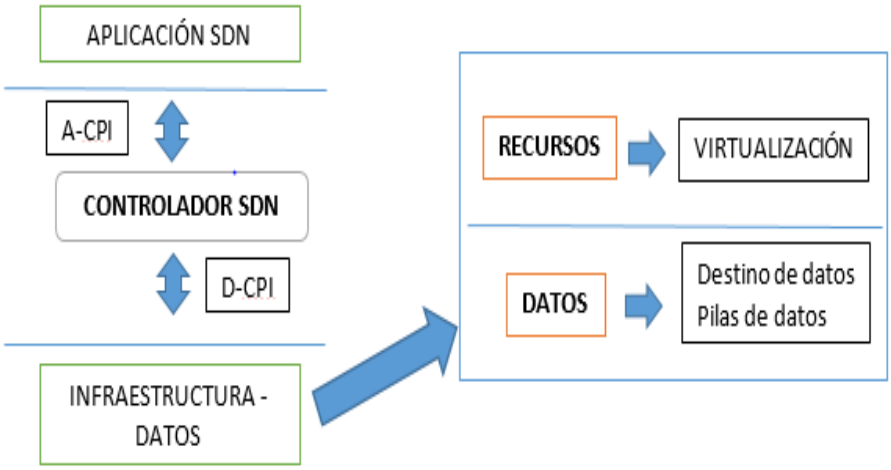


Ilustración 2 Plano de datos o arquitectura. Fuente: [19]

- **Plano de controlador.**

  El controlador SDN es una entidad de software que tiene control exclusivo sobre un conjunto de abstracto de recursos del plano del control, es decir es la entidad que controla y configura los nodos (Ilustración 3), se enfoca en la toma de decisiones y la selección del mejor camino para el tráfico según la red lo requiera[20].

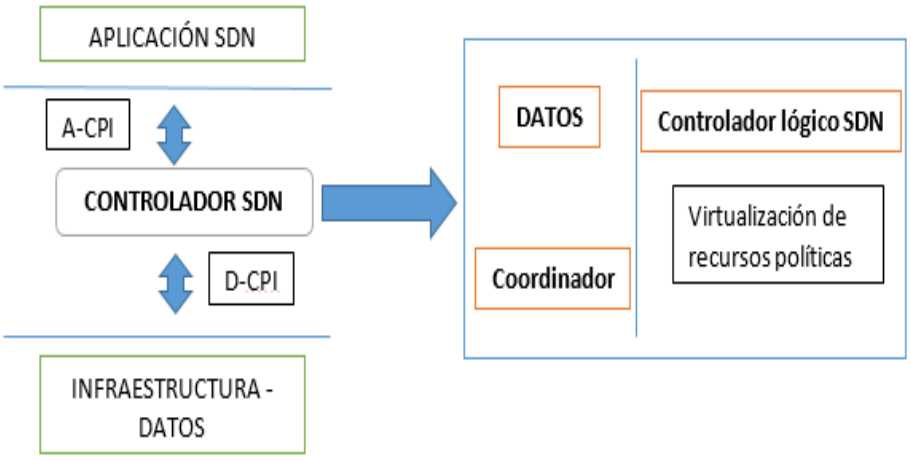


Ilustración 3 Plano de controlador. Fuente: [19]

- **Plano Aplicación.**

  Esta etapa consiste en la implementación de aplicaciones para usuarios finales, las cuales utilizan servicios de comunicación de SDN a través de API[10]. Entre las API's las que destacan están balanceo de carga, control de políticas, seguridad (Ilustración 4).

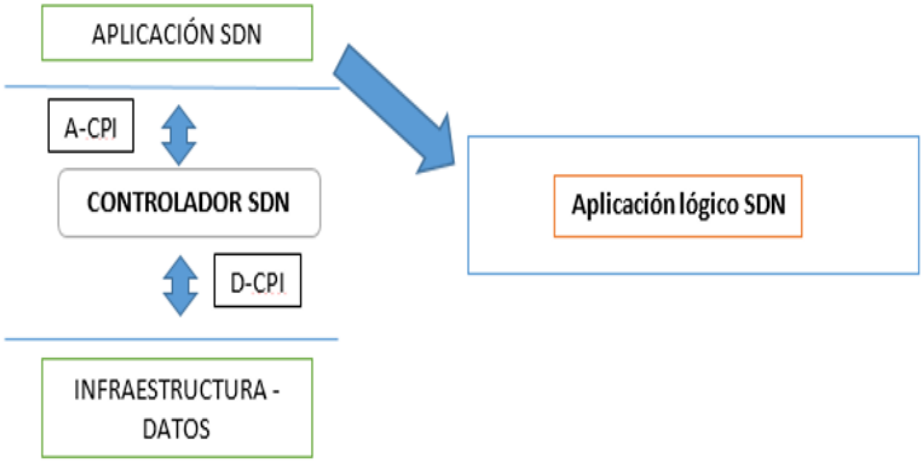


Ilustración 4 Plano de aplicación. Fuente: [19]

Protocolos de redes SDN.

El principal protocolo utilizado en SDN es Openflow pero también existen otras alternativas para estas redes que son totalmente acogidas por los usuarios ni las organizaciones.

- **OpenFlow.**

  El protocolo Openflow fue creado por miembros de las universidades de Stanford, Instituto de Tecnología de Massachusetts (MIT) y Berkeley, y constituye una de las bases principales para las redes definidas por software. Este protocolo es de lenguaje abierto, y se basa en la creación

[10] Interfaz de programación de aplicaciones

de tablas de flujos dinámicas que permiten que diferentes elementos de la red puedan comunicarse entre sí [7].

Con la popularización y el uso más frecuente de las redes definidas por software, las empresas en la actualidad están creando equipos que soporten el protocolo Openflow. En el año 2013 fue creada la asociación OpenDayLight por parte de varios fabricantes, se trata de una propuesta para crear un controlador que erradique una de las preocupaciones principales de las redes abiertas y programables como lo es la fragmentación de software y protocolos. La Linux Foundation[11] brindó todo su apoyo para realizar este proyecto [7].

OpenFlow se considera la primera interfaz para realizar la comunicación entre el plano de control y el plano de datos de una red SDN. Este protocolo permite que se pueda administrar el planto de envío de datos en dispositivos como routers y switches, esto brinda la oportunidad de tener un manejo centralizado de la red [7].

Actualmente es posible encontrar una amplia gama de dispositivos que soporten el protocolo OpenFlow, pero que se divide en dos tipos particulares al momento de utilizar el Protocolo OpenFlow.

OpenFlow ofrece varios beneficios entre los que están la programación del mismo con la intención de cambiar la estructura y el tratamiento de los datos. Otra función es el acortamiento de tiempo en cambiar de una tecnología a otra en caso de que se lo requiera, también proporciona

[11] Consorcio tecnológico enfocado al desarrollo de software en sistemas Linux

inteligencia centralizada[12], lo que permite un mejor control de los dispositivos conectados a la red.

En funciones avanzadas permite manipular que puertos estarán operando y cuál será su función, así mismo su importancia dentro de la red en función del tipo de servicio (ToS).

- **NETCONF.**

  NETCONF cumple la función de gestionar la configuración y define un procedimiento simplificado a través del cual un dispositivo de red se puede administrar, la información de datos de configuración puede ser recuperada, y los nuevos datos de configuración se pueden cargar y manejar [21].

  Este protocolo está orientado a la conexión, esto quiere decir que la conexión de red debe ser confiable y secuencial. En cuestión de recursos NETCONF facilitará el uso de los mismos cuando el servidor lo requiera y así mismo los liberará cuando detecte que no están operativos con lo que se obtiene una administración eficiente de recursos [22].

  Entre las características de este protocolo está definir parámetros de inicialización de cada dispositivo, en cuestión del orden de tratamiento de los datos utiliza FIFO[13] como mecanismo principal, además en caso de fallos de seguridad el controlador siempre anda creando puntos de restauración. En cuestión de los controladores puede soportar ONOS, OpenDayLight y RYU [23].

  Una desventaja es que cuando surgió este protocolo su implementación requería el desarrollo de un producto nuevo y que su incorporación al sistema operativo implicaba, para

[12] Hace referencia a una red adaptativa en función de los requerimientos del cliente
[13] Sistema de almacenaje que utiliza el concepto “el primero en entrar es el primero en salir”

el equipo, un consumo de recursos mayor. Por eso solo se lo encontraba en dispositivos de Cisco, Juniper y Nokia [23].

- **Protocolo de Gestión de Base de Datos Open vSwitch (OVSDB).**

  OVSDB se encarga de gestionar las operaciones de conmutación, como la creación de interfaces y la configuración de políticas para calidad de servicio (QoS). Los clústeres de gestión y control son los administradores OVSDB y el controlador OpenFlow, que pueden ser albergados en un mismo hardware o en varios [24].

  Este protocolo es orientado a la configuración de OpenFlow en función de gestionar el uso de Open vSwitch, que es un dispositivo virtual que cumple la función de switch el cual permite automatizar la red. Soporta interfaces y protocolos para administrar de forma estándar la red. El protocolo OVSDB además realiza su distribución mediante servidores físicos [25].

- **OpFlex.**

  OpFlex es un protocolo southbound desarrollado por Cisco Systems, su oferta de diseño está apoyada en los modelos declarativo y no imperativo [26]. El protocolo OpFlex de Cisco busca entre otras cosas, conservar la inteligencia de control en la infraestructura de la red en vez de agruparla en un controlador aislado, que es la naturaleza del desacoplamiento de control / reenvío de OpenFlow. Al hacer esto, OpFlex quiere mantener como dispositivos de control de una red programable a todo el hardware de la infraestructura de red, en vez de utilizar un solo servidor para el envío de paquetes hacia el controlador central SDN fundamentado en programación [27].

El protocolo Opflex fue realizado para admitir intercambiar un grupo de datos de diferentes objetos administrados que forman parte de un estándar de información. Opflex no solo usa el estándar de información, sino que puede utilizarse cualquier estándar que se base en estructura de árbol en el que todos nodos cuenten con un identificador universal de recursos (URI) [28]. Solo el controlador OpenDaylight es el que admite utilizar funciones del protocolo Opflex en su interfaz northbound [26].

- **LISP.**

  Una característica de LISP es que los dispositivos terminales[14] operan de la misma manera tal como lo hacen en la actualidad. En este protocolo se utilizan las mismas direcciones en los hosts para enviar y recibir paquetes que no presenten cambios en el camino y para el rastreo de sockets y enlaces. Cuando se habla de direcciones en el protocolo LISP[15], se hace referencia a Identificadores de punto final o conocidos por sus siglas EID [29].

  Lisp se plantea como una arquitectura que se pueda usar junto con el paradigma SDN [26], donde su propósito originalmente se basó en solucionar problemas de escalabilidad de las tablas de enrutamiento de la Zona Libre de Internet (DFZ) a través de la integración de prácticas basadas en ingeniería de tráfico en el área de los identificadores, conservando el lugar de los localizadores sin modificaciones y sumamente agregable. Actualmente el papel de LISP en SDN están presentado grandes ventajas,

[14] Hace referencia a los equipos clientes o hosts
[15] Protocolo de separación de localizador

debido a que están enfocado en estas redes y por su puuesto de un controlador como lo es OpenDayLight [30].

- **Perfil de Transporte MPLS (MPLS-TP).**

  MPLS-TP está orientado a trabajar sobre la capa red y se enfocada a redes de transporte. Los diseños IETF son la base de este protocolo y sus extensiones, con la intención de suplir aquellas necesidades que son de suma importancia para los proveedores de servicios [31].

  La ONF planteó algunas modificaciones en el protocolo MPLS orientados a OpenFlow y que puedan interactuar entre sí. Otro cambio que sugirió fue la utilización del plano fijo MPLS con una función de control simplificada basado en SDN. Al tener un plano de control simple, este se desajusta del plano en donde se encuentran los datos, las recomendaciones concebidas por la ONF tienen la intención de administrar los servicios de manera óptima [31].

  Controladores.

  El controlador es el centro de toda la arquitectura SDN, esto quiere decir que es el dispositivo que se encarga de manejar toda la lógica producida en la red y tomar decisiones de acuerdo a sucesos presentados, el controlador logra gestionar el flujo de datos mediante un conjunto de reglas producidas en las aplicaciones y las distribuye en todos los equipos que se encuentran alojados en la capa física de la red. Nos podemos referir como un sistema operativo al hablar de un controlador SDN, porque aporta en varias ramas como son la administración de la red, encontrar nuevos dispositivos al actualizar la topología, brindando servicios a la red mediante la utilización de APIs y manteniendo sesiones TCP [32].

En la actualidad existen diversos tipos de controladores destinados para las redes definidas por software, estos utilizan o admiten diferentes lenguajes de programación y plataformas, pero principalmente cumple la siguiente función: se comunican con los diferentes equipos de la red utilizando el protocolo OpenFlow. Se espera que el controlador SDN permita realizar una configuración más rápida de equipos que en las redes tradicionales, sin esperar que el fabricante realice este proceso, igualmente con el caso de aplicaciones y diferentes servicios dirigidos a estas redes [32]. Entre los controladores más importantes utilizados en SDN están los siguientes: OpenDayLight, NOX, Ryu, POX, Floodlight, Beacon, etc.

- **NOX.**

  Cuando se plantea el desarrollo de redes SDN, se requiere de un controlador que se encargue de administrar la red, lo que dio nacimiento a NOX. Un controlador basado en C que se encargaría del control, administración y desarrollo de aplicaciones enfocadas en redes definidas por software. Este controlador fue desarrollado por Nicira Networks y luego pasó a ser parte de la plataforma de virtualización conocida como VMware donde pudo desarrollarse en muchas áreas acorde a las necesidades y requerimiento [33].

  Desde su desarrollo existieron dos versiones la primera conocida como simplemente Nox y luego de su desarrollo con VMware surgió Nox Classic. Ambas versiones contaban con características y facilidades descritas en la tabla 6. En el caso de la primera versión esta no cuenta con soporte de aplicaciones de Red y Web pero si admite aplicaciones básicas como Openflow, Switch una de sus desventajas es

que solo admitía lenguaje C++ y carecía de una interfaz gráfica pero una de las ventajas más importantes es su velocidad al ejecutar los procesos al menos los básicos [33].

En su contraparte está Nox Classic que evolucionó bastante con su antecesor debido a que ya admitía aplicaciones de red para la administración así mismo contaba con servicios web y cliente-servidor. Además, contaba con una interfaz gráficas para su uso lo que hace más entendible su funcionamiento y configuración, además, cuenta admite los lenguajes C++ y Python [33].

Algunas de las características de NOX principales está que permite el uso de Api en C++ y OpenFlow en su versión 1.0. bajo entornos Linux, donde puede admitir complementos para el descubrimiento de topología, interruptor de aprendizaje y conmutador de toda la red [34].

- **POX.**

POX es un controlador desarrollado con la intención de cubrir los requerimientos en SDN usando Python en Windows. NOX fue el punto de partida para la creación del controlador POX, que es un proyecto en constante evolución y está pensando específicamente para uso exclusivo en SDN; además trabaja con lenguaje Python por lo cual es compatible con diversos sistemas operativos. En la siguiente tabla se muestran características importantes de este controlador [35]:

Beacon.

Beacon es un controlador para OpenFlow de gran velocidad, que se caracteriza por ser multiplataforma debido a que admite ser ejecutado en muchos ambientes como

servidores Linux, Windows e incluso permite la inclusión de sistemas operativos como Android, otra de sus características es que está estructurado de tal manera que sea modular y es basado en Java lo que permite operaciones basadas en eventos y subprocesos.

Además, permite tener una interfaz de usuario web para su configuración lo que garantiza que sea accesible en cualquier parte. Cuenta con un desarrollo amplio desde el 2010 lo cual le da una estabilidad debido a que está actualizándose constantemente [36].

- **Floodlight.**

Floodlight es un controlador para SDN basado en OpenFlow orientado a aplicaciones empresariales que cuenta con una licencia Apache y su lenguaje de desarrollo es Java. Entre las principales características de este controlador está principalmente en ofrecer un sistema modular enfocado a la escalabilidad con la finalidad de tener dependencias mínimas para la configuración de los dispositivos, así mismo permite la implementación de redes que admitan OpenFlow y redes que no. En cuestión de la administración de red se lo hace por segmentación por islas de conmutadores con hardware OpenFlow. Gracias a la ejecución de múltiples procesos y tiene soporte con OpenStack[16], que es una plataforma para desarrollo de aplicaciones en la nube [37].

La administración de la red la efectúa mediante su API REST[17] e interfaz WEB, también cuenta con módulos cargados inicialmente para cumplir tareas pequeñas de administración. Este controlador es de los que menos recursos

[16] Proyecto de computación en la nube enfocada a proporcionar infraestructura como servicio
[17] Interfaz que usa HTTP para la obtención de datos o generarlos en formatos XML o JSON

utiliza en comparativa de OpenDayLight, esto se demuestra en la manera que procesa las operaciones con rapidez y eficacia [38].

- **Ryu.**

  Ryu está orientado a incrementar la gestión de la red y la integración de herramientas para manejar el tráfico. Ryu dentro de sus funciones implementa interfaces de programa de aplicación (API), que ayudan a mejorar la administración tornándola personalizable dependiendo de las necesidades de la empresa o usuario [39].

  Este controlador permite la integración de varios protocolos en dirección sur para administrar dispositivos con soporte OpenFlow. Además, puede soportar las versiones más actuales de OpenFlow debido a que está en constante desarrollo [40].

- **OpenDayLight.**

  OpenDayLight es un controlador basado en una infraestructura multi-protocolo[18], escalable que garantiza alta disponibilidad para la implementación de redes SDN que cuenten con múltiples proveedores. Entre las características más importantes está que cuenta con una plataforma de abstracción de servicios lo que permite a los desarrolladores crear aplicaciones que puedan trabajar con diferentes equipos y protocolos sin inconvenientes [41]. OpenDayLight se basa en una arquitectura de cuatro áreas que se encuentran en la ilustración 5.

[18] Hace referencia a redes con tecnología MPLS

Dentro de su estructura (Ilustración 6), ODL se encarga de gestionar la red desde el plano del control hasta establecer los mecanismos necesarios para incluir ciertas API's.

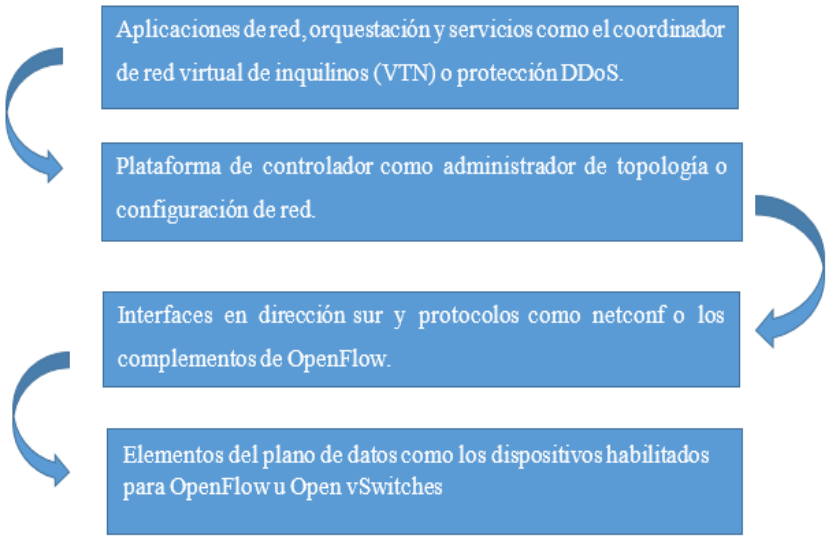


Ilustración 5 Áreas de desarrollo OpenDayLight. Fuente: [41].

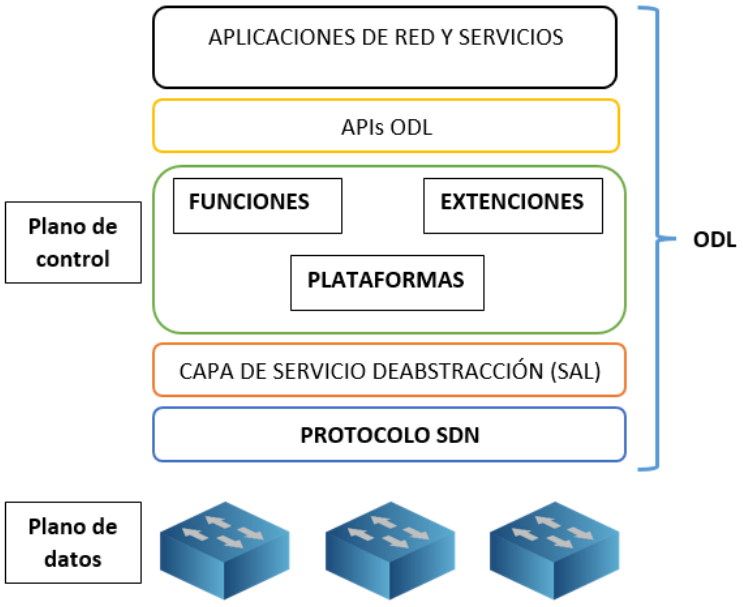


Ilustración 6 Arquitectura de ODL. Fuente: [42]

El controlador OpenDayLight se basa en las siguientes herramientas:

- **Maven:** herramienta de administración que facilita la implementación de funciones para un proyecto o de diferentes proyectos. Además, esta herramienta permite

manipular complementos y funciones que requieren los programadores para realizar modelos específicos, así lograr brindar un inicio de proyectos favorable [43].

- **Java:** es el lenguaje de programación definido para el controlador OpenDayLight, que permite crear aplicaciones y operaciones dentro del controlador. Utilizar este lenguaje brinda mayor seguridad al momento de compilar aplicaciones, otra ventaja es que brinda facilidades para implementar servicios [43].
- **Open Service Gateway Interface (OSGi):** es el back-end[19] que permite realizar toda la lógica de OpenDayLight debido a que tiene la capacidad de cargar activamente paquetes JAR (componentes de las aplicaciones), y unir varios modelos para realizar el intercambio de información [43].
- **Karaf:** se deriva de OSGi que comprende la lógica del controlador. Esta herramienta cumple la función de almacenar las aplicaciones y facilita realizar el empaquetado e instalación de aplicaciones [43].
- **YANG:** se conoce como el sitio que controla el comportamiento del controlador basándose en modelos. YANG fue creado específicamente para manipular las funciones de una aplicación y se apoya en modelos para generar APIs, que principalmente se utilizan para brindar implementaciones de servicios o aplicaciones. El módulo YANG trabaja en su mayoría con datos operacionales y configurables, como por ejemplo notificaciones y RPC [43].

OpenDayLight utiliza los siguientes módulos en su funcionamiento:

[19] Hace referencia a la separación del plano de control y el plano de datos

- **AAA:** el módulo AAA se encarga de brindar al controlador OpenDayLight seguridad en sus servicios, resguardando el acceso a todas las aplicaciones generadas a través de APIs que se encuentran en la interfaz hacia el norte o northbound [42].
- **Dlux:** es la GUI para la administración del controlador ODL (Ilustración 7), este complemento se encarga de recibir información de otros módulos intérpretes en función de los servicios de la MD-SAL. Con esto obtiene información y la ocupa para suministrar operaciones para administrar la red [44].

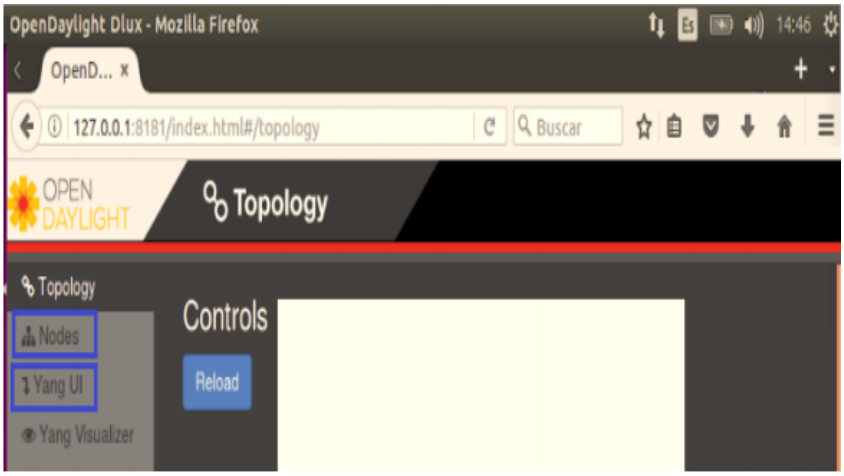


Ilustración 7 Módulo Dlux de ODL. Fuente: [44]

- **L2 Switch:** su principal función es proporcionar las funciones de un switch de nivel dos a todos los switches existentes en la topología. Realiza las operaciones de un switch tradicional, que recibe un paquete y aprende su dirección MAC[20] de origen y si su dirección de destino se encuentra en su lista de direcciones lo envía hacia el destino. En el caso de no conocer al destino, este envía un mensaje de broadcast dirigido a todos los puertos de la red. Este módulo no se

[20] Identificador único de 48 bits correspondiente a la tarjeta de red

encarga de enviar paquete de descubrimiento LLDP, solamente de paquetes Ethernet [42].

- **OpenFlow Plugin:** este mecanismo está situado en la interfaz southbound que se encarga de efectuar la comunicación entre los equipos físicos o virtuales que se encuentren en la red y el controlador para gestionar los flujos de tráfico [42].

Parámetros para la selección de controladores SDN.

A continuación, se presentan algunos parámetros para la selección de un controlador que opere en una SDN:

- **Lenguaje de programación:** elegir el lenguaje adecuado puede afectar al desempeño y rapidez con el que se desenvuelve el controlador. Algunos leguajes como Java, C ++ y Python son los que utilizan con mayor frecuencia los desarrolladores para controladores de redes definidas por software. Cada uno de estos lenguajes presentan ventajas e inconvenientes, un controlador programado en Java destaca al ser multiplataforma y tener la capacidad de subdividir aplicaciones en pequeñas partes. Por otro lado, los controladores creados con lenguaje C brinda mayor desempeño, pero no poseen una administración de memoria buena ni una GUI amigable con el usuario. Los programados en Python no pueden sobrellevar diversos subprocesos a la vez [45].
- **Soporte OpenFlow:** los administradores de red deben verificar la versión del protocolo OpenFlow que admite cada controlador y sus características, también su tendencia a soportar versiones actuales de este protocolo. Esto se debe realizar porque solo las versiones actuales del protocolo OpenFlow, como la versión 1.2 soportan el direccionamiento

IPv6, a diferencia de la versión 1.0 que no admite este tipo de direccionamiento [45].

- **Funciones del Producto:** el controlador debe cumplir con las siguientes funciones:
  - El tráfico se debe permitir de forma manual, según sea necesario dependiendo la cantidad de clientes.
  - Realizar la comunicación y conexión con switches virtuales utilizando un puerto designado para dicha tarea.
  - Al actualizar la topología el controlador debe conocer automáticamente los nuevos equipos conectados a la red.
  - Conocer la ruta de todas las conexiones que estén activas o inactivas en dispositivos de la SDN.
  - Admitir la creación de aplicaciones con contengan funciones para la SDN [46].
- **Rendimiento:** el rendimiento es un factor principal para un buen desempeño del controlador SDN, este debe manejar flujos de datos. Por ello se determinan dos elementos que forman parte del controlador SDN, que son el tiempo de organización de flujo y la cantidad de flujos por segundo. Estos elementos son muy influyentes cuando la red tiende a expandirse y agrega dispositivos y aplicaciones que son soportados por el controlador [47].

Emuladores para redes SDN.

En este apartado se estudiarán características de las herramientas pueden emular redes definidas por software:

- **Mininet.**

Mininet se encarga de la creación de redes virtuales que contienen computadores, switches y controladores SDN, su ejecución se realiza de manera rápida sobre el sistema operativo LINUX con tan solo utilizar un comando. Un

computador emulado con Mininet cumple las mismas funciones que un ordenador real, debido a que es un proyecto de código libre. Su línea de comandos es amigable con el usuario, le permite interactuar con la red virtual sin tener grandes dificultades, también, contiene una API programada en lenguaje Python para la creación y manipulación de redes por medio de líneas de comandos [32].

Entre las principales características están:

- Es una herramienta flexible que permite crear topologías de SDN con diferentes funciones, tanto en sistemas operativos Linux como Windows.
- La implementación de modelos en esta herramienta no requiere modificar el código ni las características de los equipos finales.
- Posee una interfaz en modo gráfico.
- Presenta una manipulación interactiva y en tiempo real de las topologías que se crean mediante la línea de comandos.
- Se caracteriza por crear redes escalables y sin dificultad de agregar nuevos equipos a las topologías.
- Se pueden realizar pruebas con prototipos que se encuentren en experimentación de manera simple y sencilla [32].

- **VNX.**

  VNX es una herramienta de virtualización de redes open source para todo tipo de aplicaciones con la finalidad brindar un mecanismo para la prueba de redes en entornos virtuales. VNX permite el despliegue de escenarios de red bajo topologías predefinidas por el usuario y con la facilidad de poder conectar con redes externas.

Originalmente VNX no se lo desarrolló para SDN pero en marzo del 2016, enfocó sus versiones a la integración de OpenvSwitch con soporte para configuraciones de vlan lo cual ya permitió implementar algunos escenarios de prueba enfocadas a SDN [48].

- **Estinet.**

  Es una plataforma para el desarrollo de redes, enfocada en el estudio de modelos de arquitecturas por capas como el modelo OSI, lo que diferenciaba a este emulador y simulador de redes de otros softwares es que permite interactuar en la capa física directamente. Según la página oficial del desarrollador [49], Estinet entró al desarrollo de IOT y SDN con la intención de brindar a una nueva herramienta versátil que interactúe con estos ambientes. Entre las principales características de Estinet están las siguientes:

- Integración de protocolos del Kernel de Linux para aplicaciones de capa 3 y 4.
- Compatibilidad con aplicaciones reales para su ejecución.
- Permite llevar las topologías creadas a la red con la intención de que los dispositivos emulados interactúen con los que estén conectados a la red.
- Tiene una GUI (interfaz gráfica de usuario) con la intención de ayudar a la administración de la red creada.
- Admite tanto redes cableadas como redes inalámbricas basándose en estándares como Ethernet (IEEE 802.3) y WIFI (IEEE 802.11).
- Permite generación de tráfico TCP/UDP.
- Permite mantener ambiente totalmente simulados.
- Admite modos de emulación y simulación [49].

- **GNS3.**

  Como visión general GNS3 (Simulación gráfica de redes), es un software simulador que permite diseñar topologías de redes complejas para luego de la mano de la simulación y virtualización interactuar con los equipos. GNS3 se caracteriza por la admitir más de 20 de los proveedores más importantes de equipos de telecomunicaciones en nivel mundial con lo que un administrador puede probar topologías con diferentes equipos [50]. Ente sus principales características están las siguientes:

  - Multiplataforma puede trabajar tanto en entornos Linux, Windows y Mac.
  - Diseño de infraestructuras complejas en entrono de interfaz gráfica
  - Maneja ambientes de virtualización.
  - Admite varios protocolos de enrutamiento.
  - Conexiones simultaneas con el mundo real [50].
  - Permite emular y simular equipos.

- **OMnet ++.**

  OMNeT ++ es un entorno que proporciona una arquitectura una modular que se compone de programas en C++, los cuales se ensamblan en un leguaje de alto nivel (NED). Todos los módulos están bajo licencia open source, lo que permite la integración y el desarrollo de muchas APIs en el sector de las redes. En si OMNet++ no fue diseñado para simulación de redes, pero si estructura permite su uso con bastante efectividad [51].

  Esta plataforma admite muchos componentes, pero entre los principales están:

  - Librería de Kernel de simulación.

- Lenguaje descriptivo de topología.
- Simulación de IDE fundamentado en elipse.
- Interfaz de simulación interactiva.
-
- Librería de comandos para la simulación de modelo [51].

- **DCE/ns-3.**

  El módulo DCE ns-3 brinda formas fáciles para ejecutar en ns-3 aplicaciones existentes de los protocolos de red en un área de usuario [52].

  Actualmente, se conoce que para implementar diferentes protocolos de enrutamiento se requiere agregar versiones CCNx CCN y otras recientes dentro del núcleo de la red de Linux que funciona en el emulador DCE. Lo expuesto anteriormente permite a investigadores utilizar protocolos que no tengan ningún tipo de modificaciones [52].

  Métricas de evaluación.

- **Performance.**

  Se lo define como el rendimiento de una red en función de la velocidad de transferencia al realizar alguna tarea o proceso [53]. Llevando a las redes SDN esta métrica permite ver cuáles son los recursos que utiliza cuando la red esté operativa, así mismo si su rendimiento bajo con su funcionamiento.

- **Ancho de banda.**

  Se define como la capacidad de transmisión de datos dentro de una red, relacionando tanto velocidades de subida como de bajada. Este parámetro se mide en Mbps [53].

- **Latencia.**

  Es el parámetro que permite medir el tiempo que tarda un paquete en llegar a su destino. Por lo general los paquetes dentro de una red tienen un retardo impredecible pero afecta considerablemente la calidad del servicio dentro de una red [53].

- **Rendimiento.**

  Es un atributo fundamental dentro de una red debido a que evalúa la velocidad de la red en función del software y hardware que se esté usando en la misma. Este es uno de los parámetros que más problemas causa ya que es afectado directamente por los equipos y softwares, y si estos no están actualizados pueden generar un rendimiento instable dentro de la red [53].

Protocolos de red.

Los protocolos de red son los mecanismos que especifican como transmitir la información entre varios dispositivos dentro un mismo entorno. Estos protocolos independientemente de la arquitectura están presentes, pero se dan a notar en las arquitecturas basadas en capas como el modelo de referencia OSI. Para este estudio se utilizan redes basadas en TCP/IP Y SDN. A continuación, se mencionarán algunos protocolos presentes en este estudio:

Protocolo	Descripción
UDP	Permite el envío de paquetes datagrama dentro de una red sin tener establecida una conexión. Este protocolo está vinculado a la capa

	de transporte del modelo de referencia.
ARP	Permite el descubrimiento de dispositivos de una red vinculando direcciones IP a direcciones físicas (MAC) [54].
ICMP	Este protocolo es encargado de informar sobre los errores dados durante la transmisión de los datos. Estos errores se informan a través valores numéricos [55].
LLDP	Está enfocado al descubrimiento de los dispositivos vecinos para mantener actualizada la red ante algún cambio en la misma. En la actualidad es el protocolo libre es utilizado por fabricantes sustituyendo el protocolo CDP propietario de CISCO [56].
TCP	Es orientado a la conexión, es decir permite la conexión de extremo a extremo sobre un red de internet no fiable [57].
IP	Es un protocolo no orientado a la conexión que se encarga de mover los datagramas de un punto a otro dentro de la red. Entre otras de las funciones está la de fragmentación de la información en paquetes de menor tamaño [57].

RTP	Protocolo es utilizado como medio de transporte para realizar transmisiones archivos multimedia como audio y vídeo en tiempo real. Entre sus funciones está el descubrimiento del tipo de archivo, programar un tiempo para transmitir y detectar la pérdida de paquetes [58].

Tabla 1 Protocolos orientados a la investigación
Autor: Los investigadores

Capítulo 3

Identificación de protocolos de redes SDN.

A pesar de que OpenFlow es el protocolo más utilizado en SDN también existen otras alternativas que se pueden utilizar, estos protocolos son Netconf, OVSDB, OpFlex, LISP, MPLS-TP, entre otros.

OpenFlow es el protocolo que se implementa en la actualidad en las redes SDN porque permite que el plano de control y el plano de datos se comuniquen para manejar de forma centralizada la red.

El protocolo NETCONF no se basa en asegurar la comunicación entre los planos de SDN; sino, que se enfoca en permitir la configuración de equipos de red. El protocolo OVSDB se encarga de realizar la tarea de gestión del conmutador y también de administrar implementaciones de un switch virtual Open vSwitch.

De las funciones de la tabla sobresale una ventaja del protocolo OpenFlow con respecto al resto de protocolos, que es poder comunicar el plano de control y el plano de datos; en otras palabras, permite realizar la comunicación entre todos los terminales y el controlador para gestionar los recursos de red.

En este punto se seleccionó a Openflow como protocolo para esta investigación debido a que admite muchas características para redes SDN, gracias al apoyo de las principales organizaciones impulsadoras de este prototipo de redes. Así mismo es el protocolo que ha ido evolucionando conjuntamente con las redes SDN.

Identificación de Controladores de redes SDN.

El controlador es el elemento central de las redes definidas por software, por ello se deben analizar varias características para determinar diferencias entre los principales tipos de controladores. En la tabla 11 se muestran los controladores más importantes que se usan en SDN y algunas de sus características.

En cuanto a la elección del controlador que se adapte a las necesidades del proyecto se consideraron algunas características. Estas características permitieron determinar que el primer controlador NOX no soporta versiones actuales del protocolo Openflow ni tampoco del sistema Operativo Ubuntu. No está disponible para diferentes plataformas; sino, tan solo para Linux, en sus inicios no contaba con interfaz gráfica lo que constituía una gran desventaja para controlar la red.

El controlador POX surgió como mejora de NOX, este se podía utilizar en diferentes plataformas, tales como Windows, Mac Os y Linux. Cuenta con una interfaz gráfica, pero carece de interfaz web, soporta versiones antiguas del protocolo Openflow por lo cual no es un controlador adecuado para implementaciones actuales.

Floodlight es un controlador que aporta mejoras porque cuentan con interfaz web y soporte para diferentes plataformas, está escrito en lenguaje Java, pero presenta limitaciones en sus funciones. El controlador Beacon tiene el inconveniente de no admitir soporte para versiones actuales del protocolo OpenFlow.

Los controladores Ryu y OpenDayLight se están implementando en proyectos actuales de redes SDN porque poseen gran velocidad y tienen mejor manejo de tráfico. Ambos controladores soportan versiones actuales del protocolo Openflow pero Ryu no tiene soporte multiplataforma, solo trabaja en Linux. Ryu se programa en lenguaje Python y OpenDayLight en Java, la diferencia entre estos lenguajes de programación es que Java permite ejecutar multiprocesos, pero en cambio Python tiene dificultades para manejar múltiples subprocesos.

Aunque Ryu sería una buena opción al momento de elegir un controlador existen ciertos aspectos que hacen inclinarse por OpenDayLight como la recepción de actualizaciones cada seis a nueve meses. La interfaz gráfica de OpenDayLight es amigable y tiene opciones más completas dónde se pueden ver tablas de los hosts, a diferencia de la interfaz de Ryu que es básica.

Entre estos controladores se seleccionó a OpenDayLight (ODL) debido a que es el que mejor se adapta a las necesidades de la investigación.

OpenDayLight es una plataforma de software abierto enfocado en la administración y gestión de redes SDN que está en constante desarrollo con la finalidad de poder implementar nuevas funcionalidades.

Tiene el respaldo oficial de grandes compañías tecnológicas como CISCO, Microsoft, IBM, JUNIPER NETWORKS, VMWARE, REDHAT, entre algunos más, todos estos patrocinadores están trabajando en proyectos basados en ODL y SDN. Permitiendo que este controlador cumpla con uno de los objetivos de SDN que es la integración de diversos fabricantes trabajando

bajo un mismo entorno, sin problemas de compatibilidad principalmente de protocolos.

Es multiplataforma, luego de su instalación correcta en un host o servidor Linux (Ubuntu) al tener una interfaz WEB, permite la administración en cualquier otro equipo sin problema. Lo que facilita al administrador ingresar a la configuración de los equipos sin dificultad.

Identificación de Emuladores.

Desde la aparición del concepto de SDN han ido surgiendo muchas aplicaciones que permiten interactuar con cada uno de los componentes de esta arquitectura. Estas aplicaciones brindan una herramienta práctica a investigadores y organizaciones tecnológicas para que profundicen y desarrollen nuevas soluciones bajo esta arquitectura.

Entre los candidatos para utilizar en esta investigación están Mininet, Estinet, GNS3, OMnet++ y VNX. Muchos de estos proyectos tienen características bastantes interesantes con pros y contras; además, orientados a diversas áreas de estudio.

En la tabla 12 se detallan varias características que deben tener los emuladores de redes para considerar su elección. Además, se puede visualizar que estos emuladores permiten realizar diferentes funciones para garantizar la emulación redes escalables y adaptables a cambios.

Entre las características para la selección del emulador más conveniente para la presente investigación están las siguientes: la compatibilidad con protocolos SDN como Openflow, tener la capacidad de correr controladores que

estén en desarrollo y tengan versiones estables, también se considera el lenguaje de programación que soporta o si tiene uno propio para la configuración de la capa de control.

Otro punto importante es el soporte y documentación del emulador debido a que ayuda a obtener información acerca de configuraciones, así mismo el emulador debe permitir herramientas para la generación de tráfico, captura de paquetes, evaluación del performance, entre otros. En cuestión del funcionamiento debe ser una aplicación intuitiva, de preferencia que tenga una GUI. Además, debe permitir ambientes de emulación, simulación o virtualización de equipos con la finalidad de poder interactuar con dispositivos que estén interconectados.

Características	Aplicaciones de emulación				
	Mininet	**GNS3**	**Estinet**	**OMnet++**	**VNX**
Open Source	SI	Si	Si	Si	Si
Free Software	Si	Si	No	Si	Si
Emulación	Si	No	Si	No	Si
Simulación	Si	Si	Si	Si	Si
Virtualización	Solo para host virtuales	No	No	No	Si
GUI	Si (sub aplicación Miniedit)	Si	Si	Si	Si
Multiplataforma	Solo linux (Ubuntu)	Si	Solo Linux	Si	Solo Linux
Programabilidad	Si	Si	Si	Si (contiene frameworks)	Si
Escalabilidad	Alta (multiproceso)	Alta (un proceso)	Alta (un proceso)	medio (un proceso)	Baja
Controladores soportados	OpenDaylight, Ryu, Floodlight	OpenDaylight, Ryu, Floodlight			Pox, Nox, beacon
Compatibilidad con openflow	Si	Si (complemento DCE)	Si	Si (Extención para INET)	Si(dependencia OVs)
Personalización de envío de tráfico	Si	No	No	Si	Si
Herramientas de tráfico	Wireshark, iperf	Wireshark, iperf	No especifica	SUMO	Wireshark, iperf
Leguajes soportados	Python			C++, NED	Java - Python
Compatibilidad con Dynamips, olive	NO	No	No	No	Si (con ambos)
Soporte de comunidad-fabricante	Si	Si	Si	Si	Si

Tabla 2 Comparativa de emuladores en SDN
Autor: Los investigadores

Luego de comparar cada una de estas características se concluyó que Mininet y GNS3 son los emuladores de redes que más se adaptan a las necesidades de esta investigación. GNS3 admite la simulación de dispositivos más realistas debido a que se pueden imágenes ISO de routers o switches de diferentes modelos. Mininet destaca por haber surgido junto a SDN y se creó para ayudar al desarrollo de las mismas, otras aplicaciones tuvieron que adaptarse para ofrecer esta compatibilidad.

Además, es software libre y OpenSource lo que ayuda a que tenga una comunidad que siempre se mantenga en desarrollo liberando APIs cada vez más completas, añadiendo funcionalidades como poder admitir conexiones inalámbricas para interactuar con más dispositivos. Otra ventaja que tiene es la programación del controlador debido a que se realiza bajo lenguaje Python que se caracteriza por su simplicidad al escribir código.

En cuestión del funcionamiento este admite tanto una interfaz de línea de comandos como una interfaz gráfica de usuario con su aplicación MiniEdit, así también la creación de topologías es sencilla y se enfoca en formar redes escalables y programables. Algo destacable es que permite generar y capturar tráfico a través de aplicaciones como Wireshark o/y evaluación del rendimiento con Iperf.

Además, permite crear topologías y exportar un archivo en Python cargado con los parámetros necesarios para el funcionamiento en la terminal de Mininet. Este archivo con extensión *.py* se puede modificar para agregar más código que permita una mayor interacción con los dispositivos así también pruebas en diferentes ambientes.

Mininet y GNS3 admiten varios controladores como OpenDayLight, Ryo y Floodlight, lo que permite explotar las características de cada uno de estos en función de las necesidades de cada proyecto. El punto negativo que Mininet tiene es que no admite compatibilidad con fabricantes como Cisco, Juniper, Hp, Huawei entre otros, como el caso de VNX y la otra desventaja es que no es multiplataforma ya que solo se ejecuta en entornos Linux específicamente en la distro Ubuntu.

Diseño e implementación de escenarios prácticos para evaluar el desempeño de redes SDN frente a las redes basadas en arquitectura TCP/IP.

En esta etapa se encuentran los resultados de los escenarios implementados en la arquitectura SDN y en TCP/IP.

Para esta etapa se planteó un entorno de pruebas para ejecutar cada uno de los escenarios propuestos para esta investigación, el mismo se lo realizó bajo el sistema operativo Ubuntu Server en su versión 18.04 de 64bits.

En este escenario se propuso dos topologías las cuales contaban con cinco switch para el caso de SDN compatibles con OpenFlow y para TCP/IP switch de capa dos. Una vez armadas las topologías se realizaron pruebas con la herramienta jperf para generar tráfico UDP y medir el ancho de banda que se alcanzaría en cada una de las redes. Estas pruebas de ancho de banda y jitter fueron ejecutadas cincuenta veces con la intención de obtener datos precisos.

En el primer escenario se comprobó cual es el máximo ancho de banda de contiene la red de extremo a extremo. La topología utilizada en este escenario para SDN es de tipo

lineal, consta de cinco switches, dos ordenadores ubicados en los extremos de la red y un controlador (Ilustración 9).

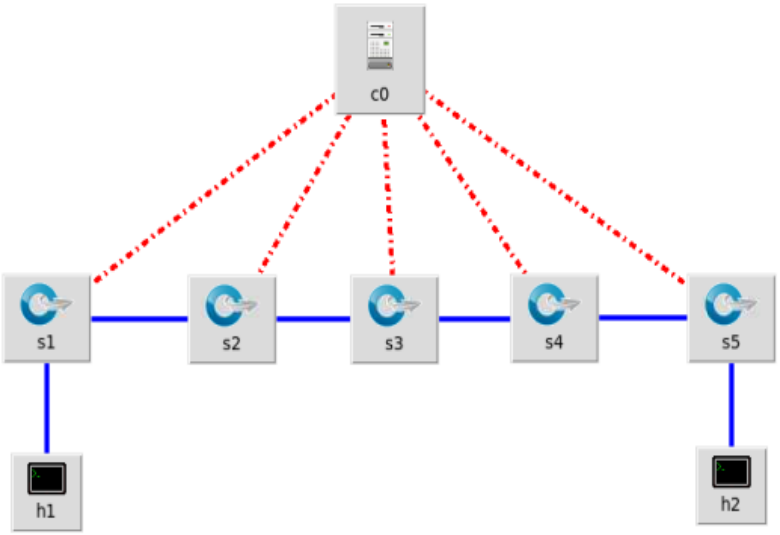


Ilustración 1 Topología SDN del escenario uno.
Autores: Los investigadores.

Dentro de la carpeta custom se inicia el escenario programado en un script de Python denominado *escenarioBW_SDN.py* y se agregan los diferentes dispositivos que se encuentran en la topología (Ilustración 10).

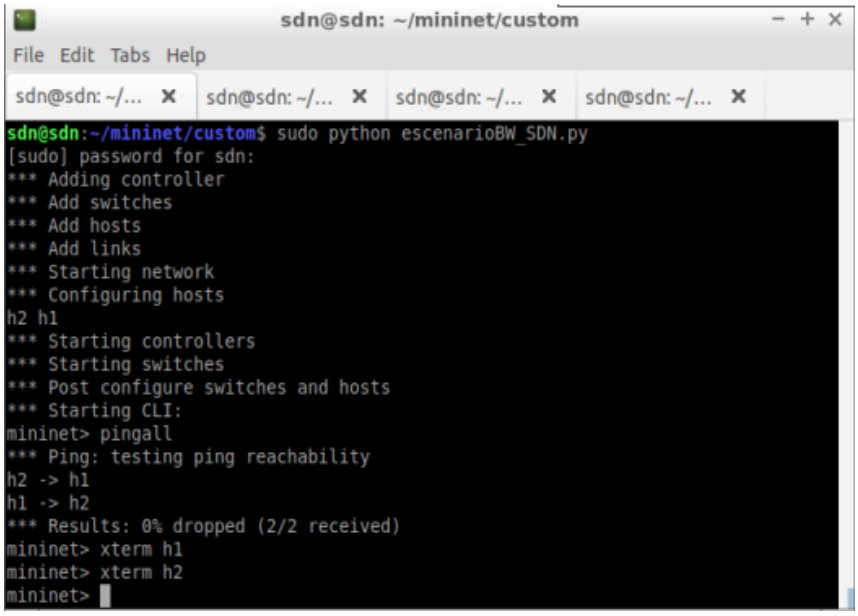


Ilustración 2 Inicio de script de la topología SDN en escenario uno.
Autores: Los investigadores.

Para medir el ancho de banda de la red se utilizó una aplicación denominada JPerf que se encarga de medir el rendimiento que tiene la red mediante el ancho de banda que puede soportar. Para empezar a medir el ancho de banda se abren las consolas de los ordenadores y se ingresa a la carpeta *JPerf-2.0.2* y se ejecuta la aplicación mediante el comando *./jperf.sh* (Ilustración 11).

Ilustración 3 Consola de los host del escenario uno.
Autores: Investigadores.

La configuración de la aplicación es bastante sencilla, en el host cliente se coloca la dirección IP de servidor y se elige como protocolo de envío a UDP porque nos permite inundar la red con tráfico para encontrar el ancho de banda para el mejor rendimiento. Para la topología de la red SDN el ancho de banda óptimo de extremo a extremo es de 112 MegaBytes/sec (Ilustración 12), con este ancho de banda no existe pérdida de ningún datagrama y llegada de los mismos fuera de orden.

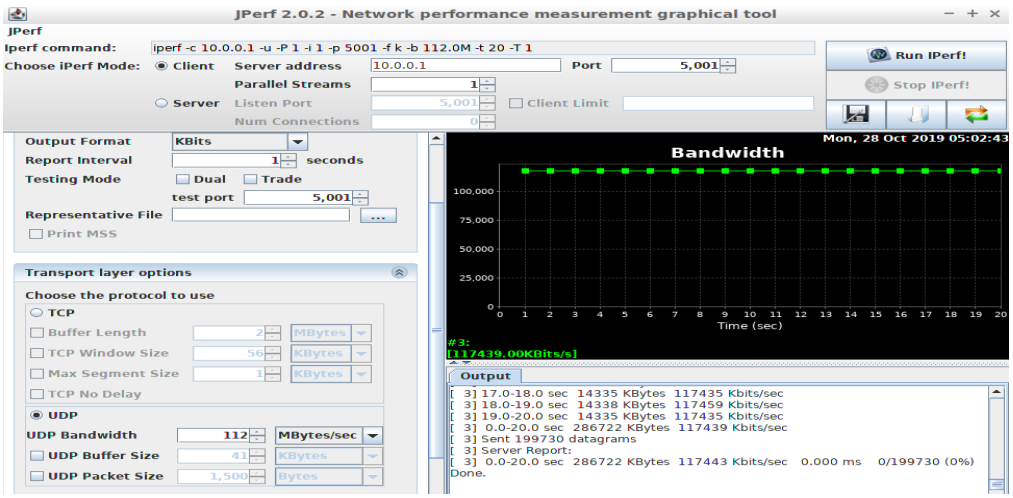


Ilustración 4 Jperf en modo cliente de la topología SDN del escenario uno.

Autores: Los investigadores.

En la ventana del host cliente aparece un gráfico en dónde se observa que es ancho de banda (Ilustración 13) es constante en el tiempo y no existe variaciones ni pérdidas de datagramas.

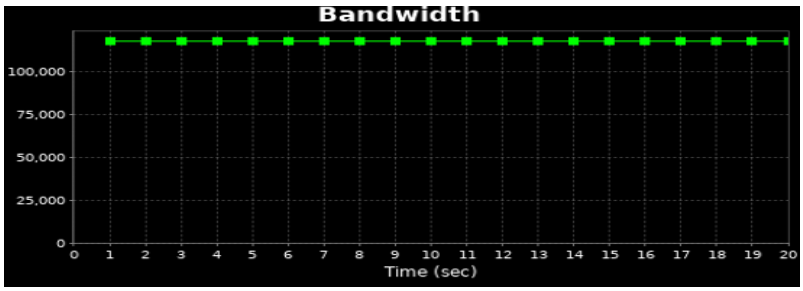


Ilustración 5 Gráfico del ancho de banda de la topología SDN del escenario uno.
Autores: Los investigadores.

En el modo servidor se coloca el puerto que es el 5,001 y de igual manera se escoge el protocolo de transmisión y se empieza a ejecutar la aplicación (Ilustración 14).

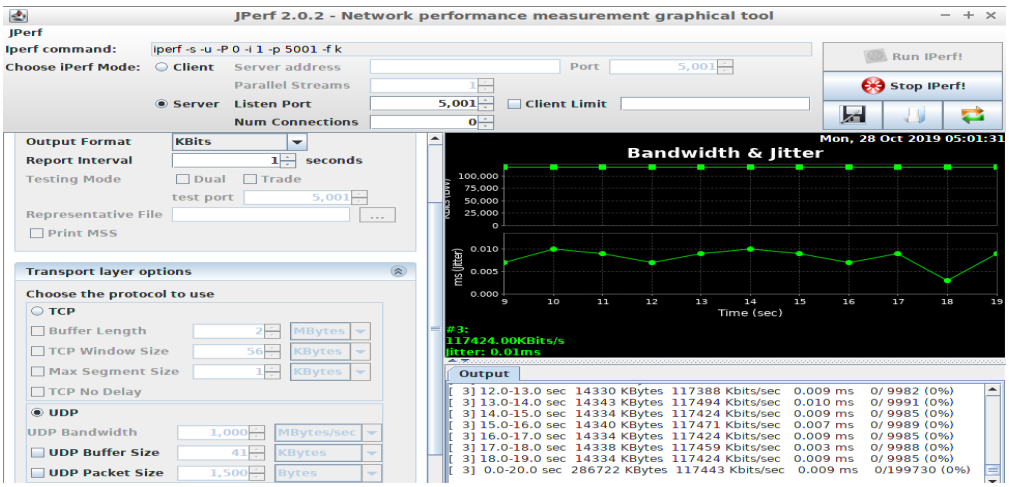


Ilustración 6 Jperf en modo servidor de la topología SDN del escenario uno.
Autores: Los investigadores.

En el servidor se encuentran gráficas sobre el ancho de banda alcanzado y el jitter que se generó en esa transmisión. Los valores de jitter son casi despreciables no sobrepasan los 0,010 ms (Ilustración 15).

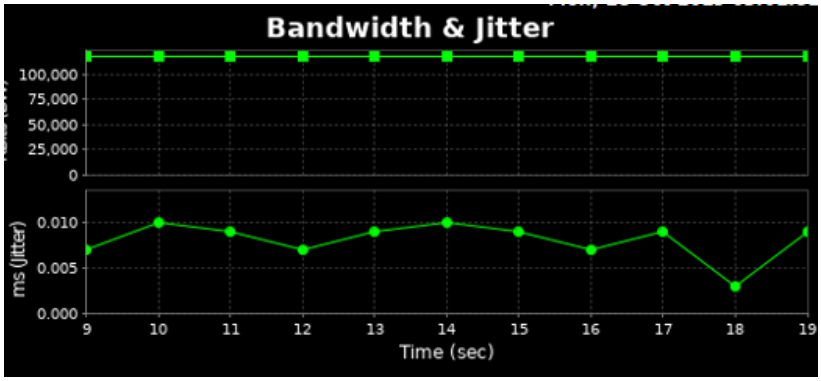


Ilustración 8 Gráfico del ancho de banda y jitter de la topología SDN del escenario uno.
Autores: Los investigadores.

Escenario Uno con arquitectura TCP/IP.

La topología utilizada en la arquitectura TCP/IP consta de cinco switches y dos ordenadores conectados a los extremos (Ilustración 16). En este caso se evaluará la misma métrica que en la topología con arquitectura SDN que es el ancho de banda, mediante la aplicación JPerf.

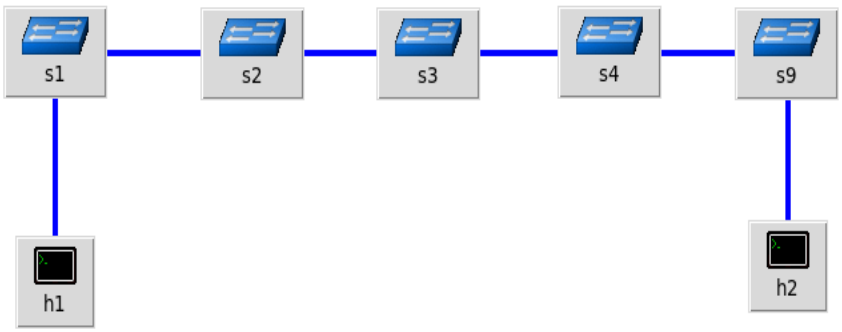


Ilustración 8 Topología TCP/IP del escenario uno.
Autores: Los investigadores.

Para ejecutar la topología simulada en Mininet colocamos el comando *Sudo Python escenarioBW.py* (Ilustración 17).

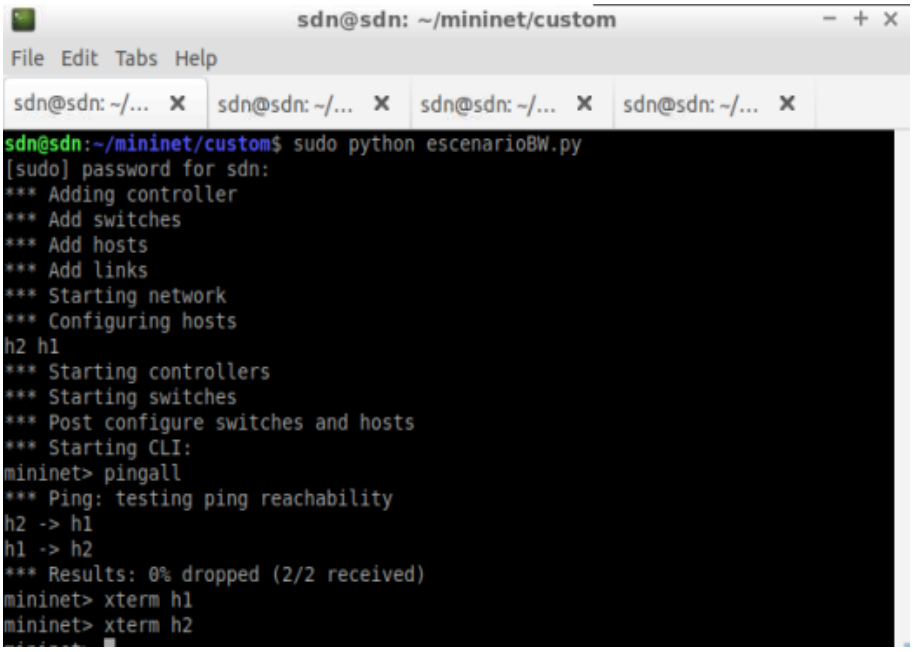


Ilustración 9 Inicio de script de la topología TCP/IP en escenario uno.
Autores: Los investigadores.

En el host que ocupa el lugar de cliente se configura la dirección IP del servidor, el puerto y el protocolo que se va a utilizar (Ilustración 18).

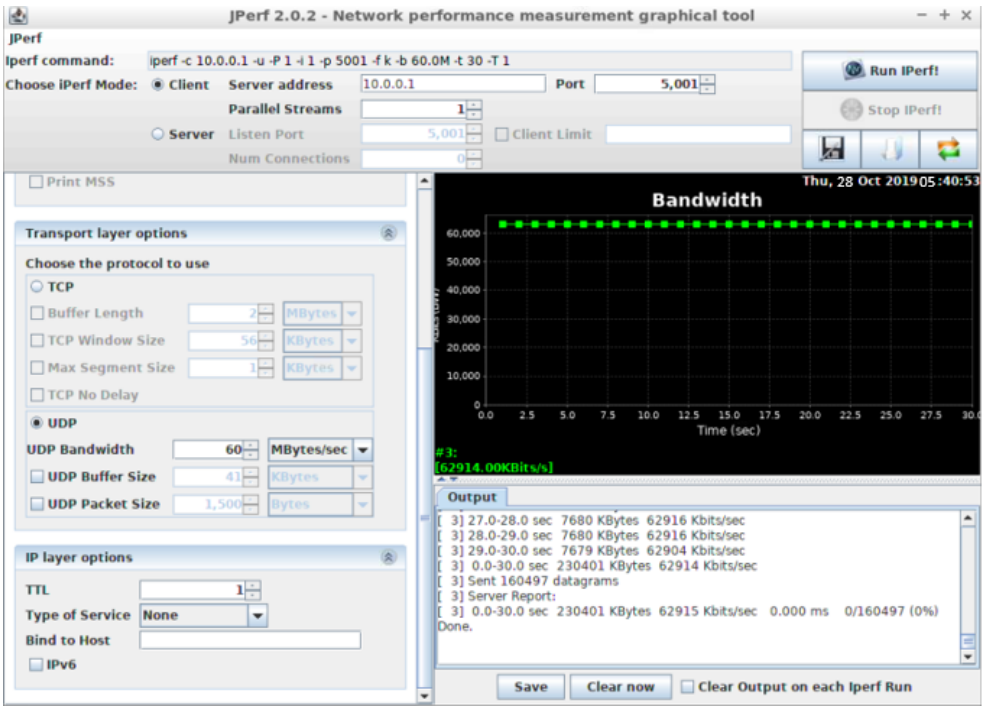


Ilustración 10 Jperf en modo cliente de la topología TCP/IP del escenario uno.
Autores: Los investigadores

En el servidor se especifica el puerto a utilizar y el protocolo, esto nos da como resultado dos gráficas, una para el ancho de banda y otro para el Jitter (Ilustración 19).

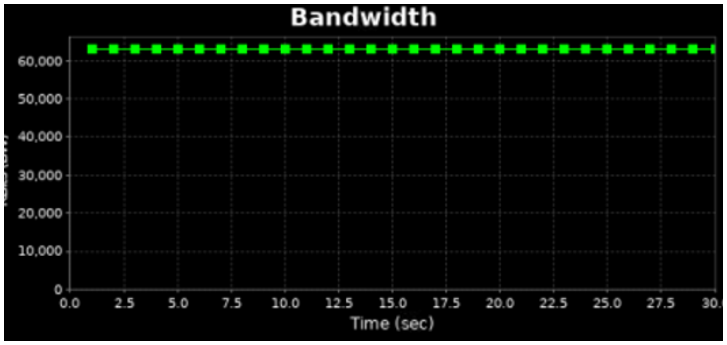


Ilustración 11 Gráfico del ancho de banda de la topología TCP/IP del escenario uno.
Autores: Los investigadores.

En este escenario se obtuvo como resultado que el ancho de banda óptimo en esta topología de red con arquitectura TCP/IP es de 60 MegaBytes/sec, en la ilustración 20 se puede observar que la transmisión se mantiene constante para este valor de ancho de banda. Adicionalmente no presenta datagramas fuera de orden ni perdidos.

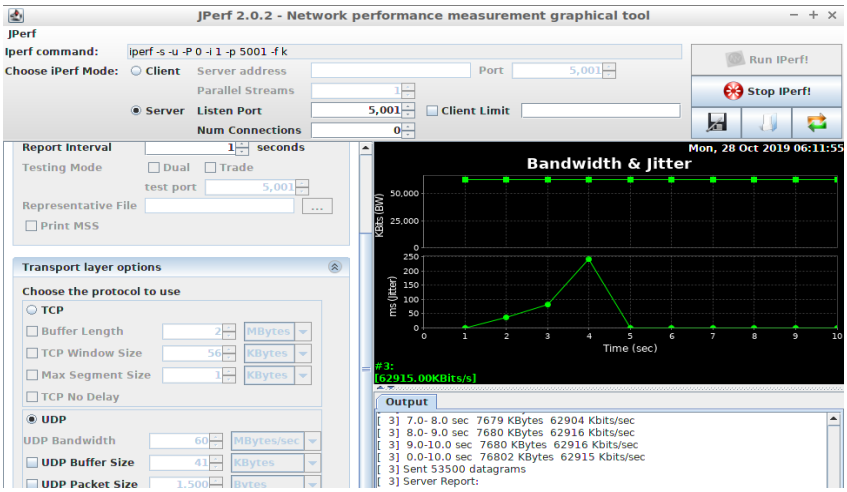


Ilustración 12 Jperf en modo servidor de la topología TCP/IP del escenario uno.
Autores: Los investigadores.

En cuanto al jitter, se tuvo un pico que superó los 200 ms para luego estabilizarse a valores muchos más bajos (Ilustración 21).

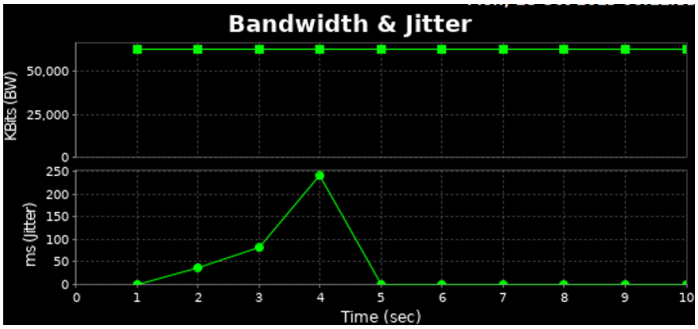


Ilustración 13 Gráfico del ancho de banda y jitter de la topología TCP/IP del escenario uno.
Autores: Los investigadores.

Descripción del escenario Dos.

En el segundo escenario se evaluó la latencia y el jitter que existe en la conectividad entre el host h1 y el h3, para esto se realizaron cinco pruebas con el comando *ping*. Las pruebas consistieron en enviar una serie de paquetes entre dos hosts de las topologías. El número de paquetes enviados en las pruebas fueron 100, 300, 600, 900 y 1000. Cada prueba se repitió veinte veces, porque los datos obtenidos no son constantes, sino que varían cada vez que se envían los paquetes.

En esta etapa se planteó un escenario que consta de una topología que se compone un controlador, cuatro switches conectados entre sí y tres hosts como se puede observar en la ilustración 22. Para medir los parámetros de latencia y jitter que se produce en el envío de paquetes entre hosts de la red.

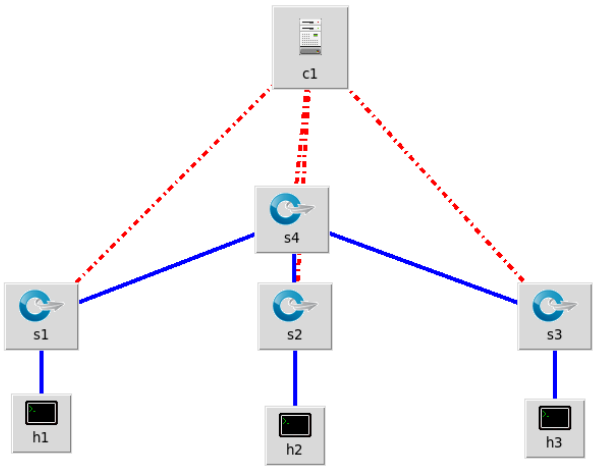


Ilustración 14 Topología de escenario dos con controlador SDN y switch OpenFlow
Autor: Los investigadores

La red se programó en un script con lenguaje Python (ver código en anexos) y se ejecutó mediante el comando *Sudo python escenarioSDN.py*. *Esto* permitió simular la red y para verificar que exista conexión entre hosts se utilizó el comando *pingall* (Ilustración 23).

```
sdn@sdn:~/mininet/custom$ sudo python escenarioSDN.py
*** Adding controller
*** Add switches
*** Add hosts
*** Add links
*** Starting network
*** Configuring hosts
h1 h2 h3
*** Starting controllers
*** Starting switches
*** Post configure switches and hosts
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
```

Ilustración 15 Creación de la topología de escenario dos mediante consola
Autor: Los investigadores

En el proceso de evaluación de la red se realizaron cinco pruebas con veinte repeticiones cada una, a través del comando *ping* entre el nodo h1 y h3. Se envió paquetes de

datos con tamaño de 65507 Megabytes más las cabeceras de encapsulación ICMP e IP. Este tamaño de paquete es el máximo admitido por el simulador Mininet. Cada una de las pruebas fue gradualmente aumentando la cantidad de paquetes enviados, el número de paquetes enviados en las pruebas fueron: 100, 300, 600, 900 y 1000 paquetes (Ilustración 24 y 25).

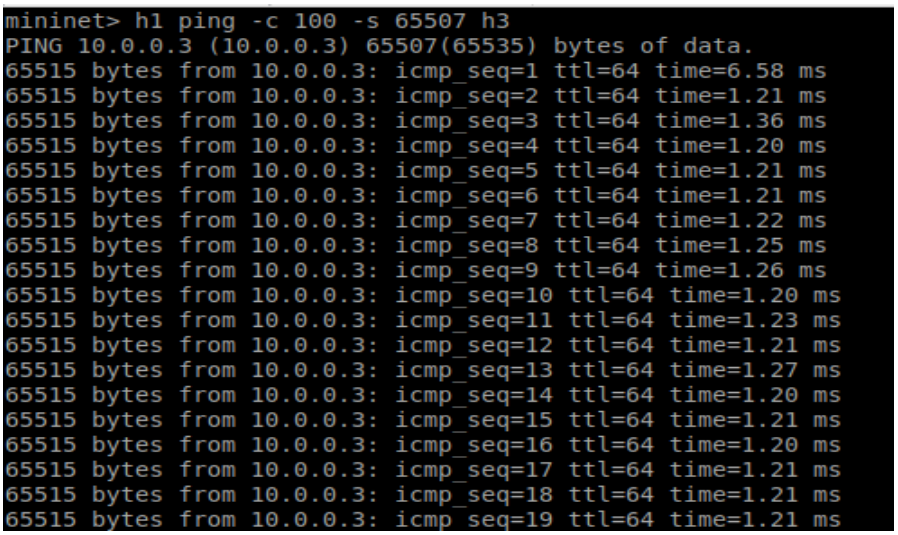


Ilustración 16 Primera prueba de ping entre el nodo h1 y h3.
Autor: Los Investigadores

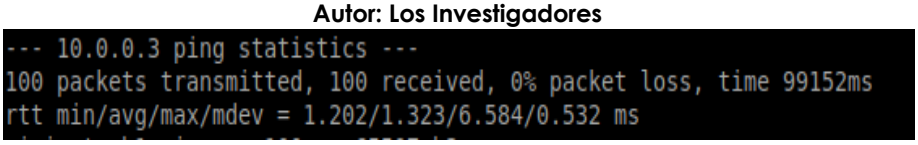


Ilustración 17 Resultado de enviar 100 paquetes entre el nodo h1 y h3.
Autor: Los investigadores

Siguiendo con el proceso, una vez realizadas las pruebas anteriores se accedió al controlador para monitorear y revisar las estadísticas captadas por OpenDayLight. Una vez dentro de la de ODL se pudo se visualizar la red creada en la opción *Topology* (Ilustración 26).

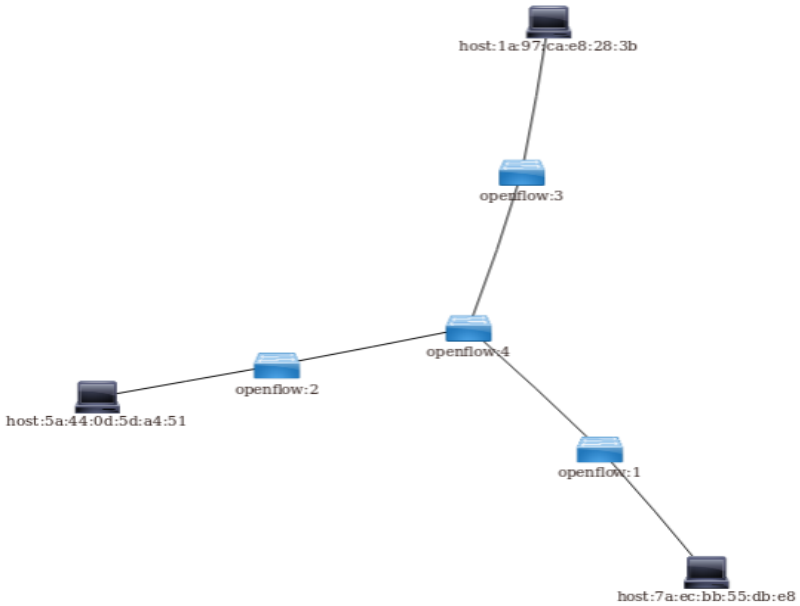


Ilustración 18 Vista de la topología creada desde la interfaz de ODL
Autor: Los investigadores

Luego en el menú de OpenDayLight en la opción de *Nodes* se observó tablas en dónde se encontraban los nodos de la red y las conexiones de cada uno; además, contenían las estadísticas del número de paquetes y bytes transmitidos entre cada uno de los nodos durante las pruebas. Esto se muestra en las ilustraciones 27 y 28.

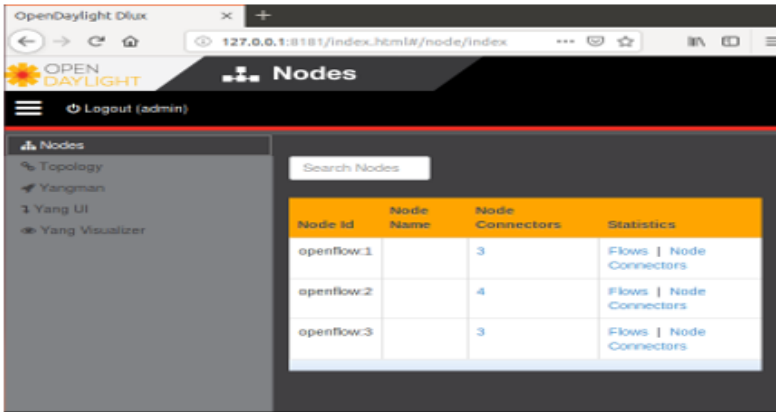


Ilustración 19 Visualización de los nodos desde ODL.
Autor: Los investigadores

Node Connector Statistics for Node Id - openflow:1

Node Connector Id	Rx Pkts	Tx Pkts	Rx Bytes	Tx Bytes	Rx Drops	Tx Drops
openflow:1:1	2203408	2218381	3278623908	3279907098	0	0
openflow:1:2	2218322	2218168	3279899541	3279881851	0	0
openflow:1:LOCAL	0	0	0	0	0	0

Node Connector Statistics for Node Id - openflow:2

Node Connector Id	Rx Pkts	Tx Pkts	Rx Bytes	Tx Bytes	Rx Drops	Tx Drops
openflow:2:1	197187	14809	272237965	1261574	0	0
openflow:2:2	42	197246	3036	272245522	0	0
openflow:2:LOCAL	0	0	0	0	0	0

Node Connector Statistics for Node Id - openflow:3

Node Connector Id	Rx Pkts	Tx Pkts	Rx Bytes	Tx Bytes	Rx Drops	Tx Drops
openflow:3:LOCAL	0	0	0	0	0	0
openflow:3:1	2203409	2218394	3278623978	3279908162	0	0
openflow:3:2	2218334	2218183	3279900535	3279883111	0	0

Node Connector Statistics for Node Id - openflow:4

Node Connector Id	Rx Pkts	Tx Pkts	Rx Bytes	Tx Bytes	Rx Drops	Tx Drops
openflow:4:1	2218160	2218314	3279881171	3279898861	0	0
openflow:4:2	14794	197171	1260299	272236620	2	0
openflow:4:3	2218161	2218312	3279881241	3279898665	2	0
openflow:4:LOCAL	0	0	0	0	0	0

Ilustración 20 Estadísticas de los nodos y el controlador ODL
Autor: Los investigadores

El resultado de la latencia se obtiene con el parámetro RTT que es el tiempo que tardan los paquetes de ida y vuelta; en la tabla 13 se encuentra el promedio de las múltiples pruebas realizadas.

N° paquetes enviados en 20 repeticiones	Latencia min	Latencia media	Latencia máx.	Jitter
100	1.212	1.351	2.462	0.227
300	1.231	1.360	4.005	0.379
600	1.194	1.357	4.391	0.372
900	1.192	1.353	4.538	0.356
1000	1.193	1.354	4.100	0.302

Tabla Resultados de latencia y jitter en red SDN.
Autor: Los investigadores

En la tabla 13 se puede observar los resultados de las diez cinco realizadas en el escenario dos, en total se enviaron 2900 paquetes en cada repetición de las pruebas entre los nodos h1 y h3. Dónde se obtuvo datos sobre la latencia máxima, media y mínima; además, el parámetro que refleja el tiempo en promedio del jitter en cada intervalo de envío de paquetes.

Escenario Dos con arquitectura TCP/IP.

En el segundo escenario para la red con arquitectura TCP/IP se replicó la topología propuesta anteriormente, pero se optó por evitar el uso del controlador, con el fin de que sea una red tradicional. La topología se la puede visualizar en la ilustración 29, en este caso también se cambió el Switch por un Legacy Switch el cual no admite el protocolo OpenFlow ni auto aprendizaje.

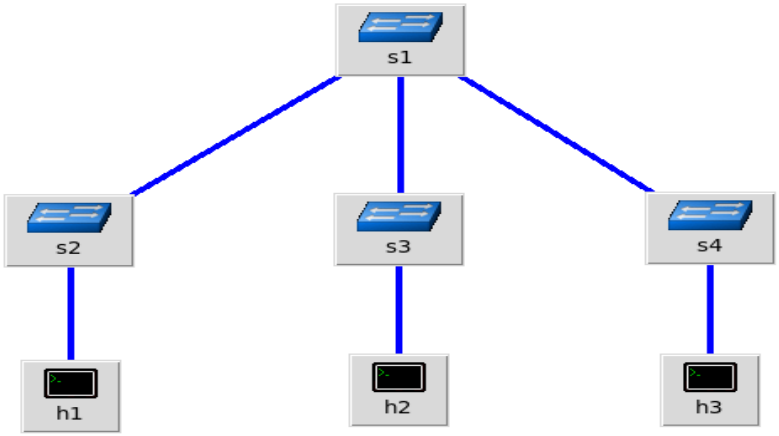


Ilustración 21 Topología de escenario dos con arquitectura TCP/IP.
Autor: Los investigadores

Luego del diseño de la red en MiniEdit se generó el script (Ilustración 30) en Python de la topología creada para ejecutarla en la línea de comandos. Donde se visualizó la creación de cada uno de los dispositivos con sus respectivas interfaces y adicionalmente se ejecutó el comando *pingall* con la intención de verificar las conexiones entre los hosts.

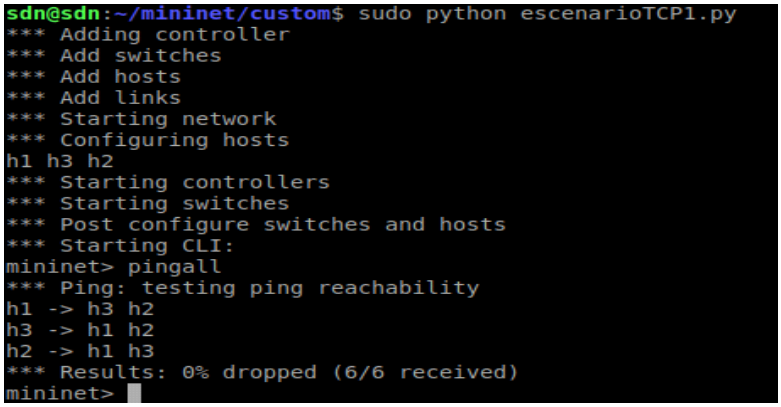


Ilustración 9 Ejecución de script en Python de Escenario 1 TCP/IP
Autor: Los investigadores

Como siguiente paso, se realizaron las mismas pruebas que la topología con arquitectura SDN del escenario dos, se ejecutaron cinco pruebas como se puede visualizar en las ilustraciones 31 y 32, en las que se iban generando cantidades mayores de tráfico hasta alcanzar los 1000 paquetes. Con esto se recopiló resultados de la conexión como valores de latencia y jitter.

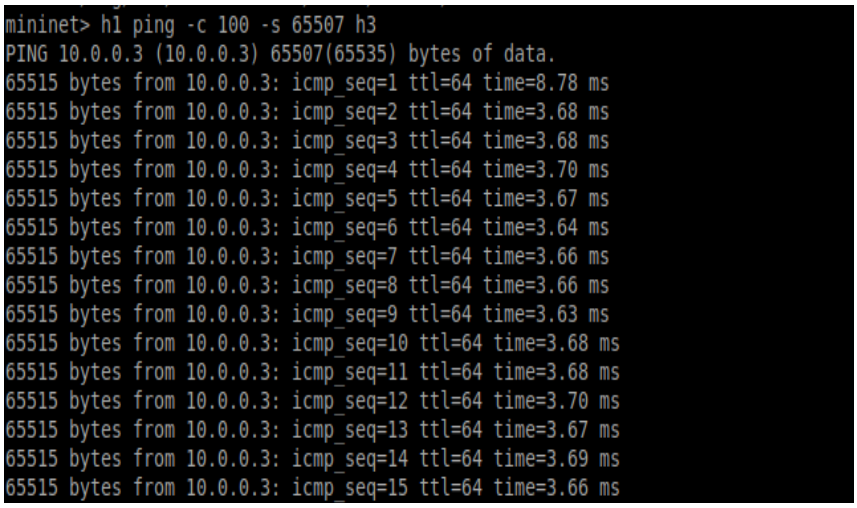


Ilustración 23 Ping con envío de 100 paquetes red TCP/IP.
Autor: Los investigadores

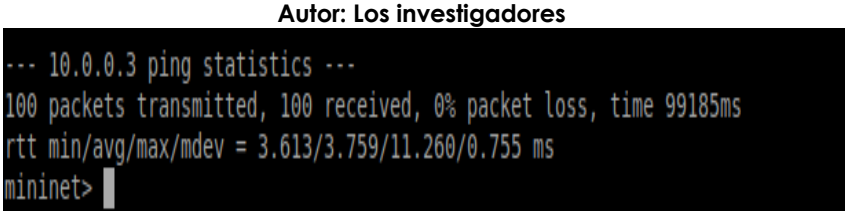


Ilustración 24 Resultado de envío de 100 paquetes en red TCP/IP.
Autor: Los investigadores

En este escenario planteado bajo arquitectura TCP/IP se evaluó la conectividad entre el nodo h1 y el nodo h3, mediante la generación de tráfico a través del comando *ping*.

Los promedios de latencia, así como de Jitter de cada una de las pruebas repetidas cincuenta veces se encuentran en la siguiente tabla:

N° paquetes enviados en 20 repeticiones	Latencia min	Latencia media	Latencia máx.	Jitter
100	3.624	3.734	10.072	1,930
300	3.614	3.695	10.312	1,051
600	3.598	3.687	10.685	1,209
900	3.589	3.682	10.193	0,838
1000	3.594	3.684	10.695	0,620

Resultados de pruebas de latencia en Escenario dos arquitectura TCP,
Autor: Los investigadores

Comparación estadística de resultados del escenario Dos.

Los resultados obtenidos en puntos anteriores ayudaron a obtener una media entre los valores de latencia de la topología con arquitectura SDN, estos valores están presentes en la tabla 15. En esta tabla se puede observar que los valores de latencia mínima, media y máxima presentan valores diferentes.

Estadísticas de grupo					
	Tipos de redes	N	Media	Desviación estándar	Media de error estándar
Latencia mínima	SDN	5	1,20440	,017009	,007607
	TCP/IP	5	3,60380	,014670	,006560
Latencia media	SDN	5	1,35500	,003536	,001581
	TCP/IP	5	3,69640	,021594	,009657
Latencia máxima	SDN	5	3,89920	,831714	,371954
	TCP/IP	5	10,39140	,285507	,127683

Estadísticas de escenario dos SDN y TCP/IP.
Autores: Los investigadores

La ilustración 33 muestra los datos tabulados de las pruebas comparando las dos arquitecturas lo cual brinda una visión del funcionamiento de las mismas con y sin el controlador.

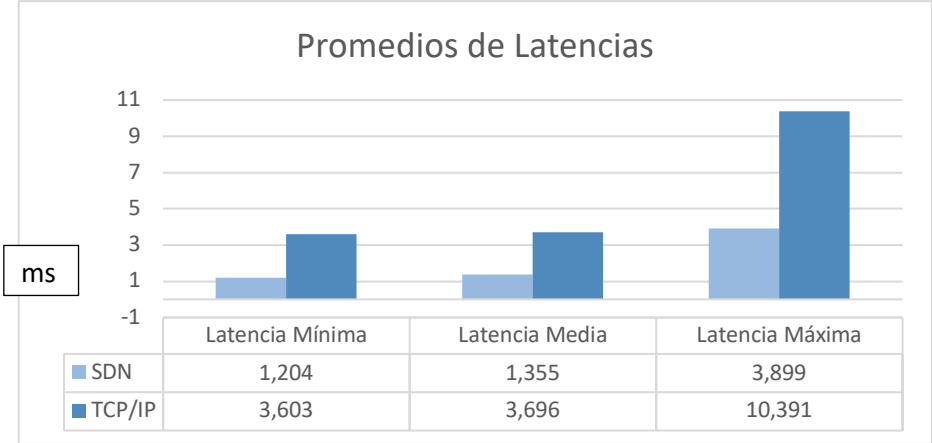


Ilustración 25 Gráfico comparativo de latencias entre arquitectura SDN y TCP/IP.
Autores: Los investigadores.

El análisis de los datos se lo realizó en el software estadístico RStudio, en dónde se analizaron los valores de latencia tanto de SDN como de TCP. Primero se realizó la prueba denominada Shapiro Wilk para verificar la normalidad de los datos (tabla 16).

RESULTADOS DE LA PRUEBA SHAPIRO WILK				
	Latencia Mínima	Latencia Media	Latencia Máxima	Jitter
SDN	p-value = 0.0877	p-value **= 0.8441**	p-value = **0.05975**	p-value = **0. 2233**
TCP	p-value = 0.5084	p-value = 0.02537	p-value = **0.2693**	p-value = **0. 1596**

Tabla Resultados de la prueba Shapiro Wilk del escenario dos.
Autores: Los investigadores.

Los valores en negrita de la tabla 16 son los que presentan una distribución normal con respecto a los valores estadísticos estándar, que suponen que si el valor de p es mayor a 0,05 se presenta una normalidad en los datos. En el resultado de la prueba Shapiro Wilk solo un valor no alcanzó el rango para estar dentro de los valores normales de distribución.

En el software RStudio también se utilizó el procedimiento estadístico "T" para muestras independientes, que define si existe una diferencia significativa entre latencias de las redes SDN y TCP/IP.

Si el valor estadístico de "p" es mayor a 0,05 se supone que no existe diferencia significativa entre los valores que se comparan, y si es menor a 0,05 sí existe una diferencia significativa entre las madias analizadas.

A continuación, se muestra tabla 17 con los resultados de la prueba T para muestras independientes para cada latencia.

Para este caso comparativo entre dos redes se determinó que existen varianzas y medias diferentes para los valores de latencia mínima, media y máxima. En la ilustración 34 se muestra la diferencia entre el jitter de las redes SDN y TCP/IP en cada una de las pruebas realizadas.

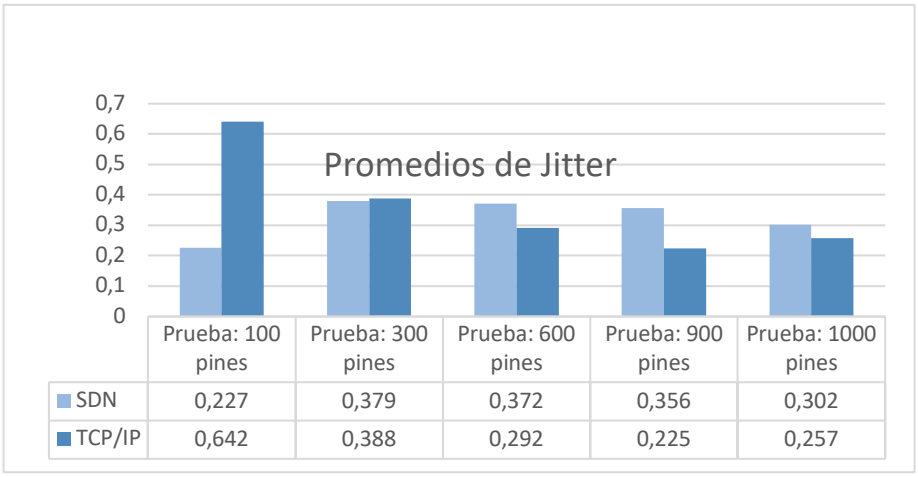


Ilustración 26 Gráfico de Jitter en SDN y TCP/IP del escenario dos.
Autores: Los investigadores.

Para las pruebas de jitter también se analizaron estadísticamente los valores, en la tabla 18 se puede ver a detalle los valores de la media obtenida. En la prueba T para variables independientes se obtuvo un valor mayor a 0,05, por lo cual se supone que no existe una diferencia significativa entre las medias del jitter de las dos redes propuestas, esto quiere decir que el jitter es estadísticamente igual (Tabla 19).

Tercer Escenario.

Descripción del Tercer Escenario.

Para la realización de este escenario se utilizó GNS3 como simulador y emulador de red, con la finalidad de llevar las pruebas en equipos reales en entornos simulados. Para este escenario se planteó 2 topologías las cuales se orienta a las arquitecturas de SDN y TCP/IP. Entre las pruebas realizadas se transmitió un vídeo con calidades distintas (HD y FHD), entre un host que actuaría como servidor y dos hosts como clientes de la transmisión. Adicionalmente, para la captura de tráfico durante la transmisión se utilizó en sniffer Wireshark. En cuestión del diseño de las topologías se determinó el uso del

emulador de red GNS3 para realizar las diferentes pruebas tanto en arquitecturas SDN como TCP/IP.

En cuestión de las pruebas de capturas realizadas en este escenario, se las realizó 30 veces donde se transmitió tanto el vído en HD como el vídeo en FHD

Topología de red con arquitectura SDN.

La topología planteada cuenta con dos router mikrotik los cuales se integró la funcionabilidad de un switch OpenFlow y la administración de los mismos mediante la herramienta Winbox. Además, para la virtualización de los hosts se utilizó virtualBox bajo sistemas operativo Ubuntu. Finalmente, para la captura de los paquetes enviados entre los diferentes dispositivos se utilizó el sniffer Wireshark (Ilustración 35).

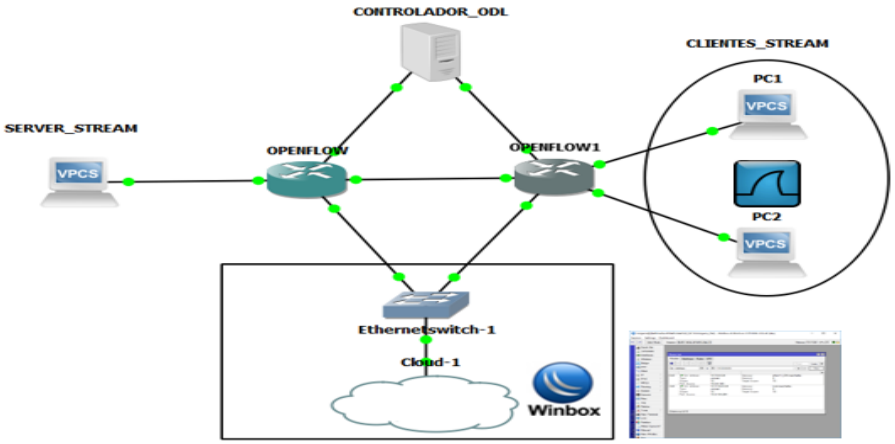
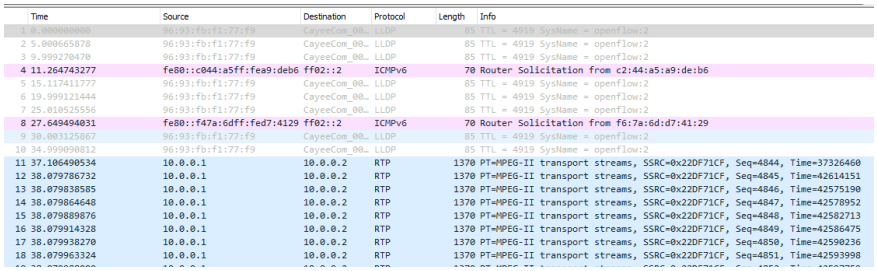


Ilustración 27 Topología de red con arquitectura SDN en GNS3.
Autores: Los investigadores.

Dentro de las pruebas realizadas en esta topología estuvo la captura de tráfico por parte del sniffer las cuales se las puede ver a detalle a continuación en la ilustración 36.



	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	96:93:fb:f1:77:f9	CayeeCom_00…	LLDP	85	TTL = 4919 SysName = openflow:2
2	5.000065878	96:93:fb:f1:77:f9	CayeeCom_00…	LLDP	85	TTL = 4919 SysName = openflow:2
3	9.999270470	96:93:fb:f1:77:f9	CayeeCom_00…	LLDP	85	TTL = 4919 SysName = openflow:2
4	11.264743277	fe80::c044:a5ff:fea9:deb6	ff02::2	ICMPv6	70	Router Solicitation from c2:44:a5:a9:de:b6
5	15.117411777	96:93:fb:f1:77:f9	CayeeCom_00…	LLDP	85	TTL = 4919 SysName = openflow:2
6	19.999121444	96:93:fb:f1:77:f9	CayeeCom_00…	LLDP	85	TTL = 4919 SysName = openflow:2
7	25.010525556	96:93:fb:f1:77:f9	CayeeCom_00…	LLDP	85	TTL = 4919 SysName = openflow:2
8	27.649494031	fe80::f47a:6dff:fed7:4129	ff02::2	ICMPv6	70	Router Solicitation from f6:7a:6d:d7:41:29
9	30.003125867	96:93:fb:f1:77:f9	CayeeCom_00…	LLDP	85	TTL = 4919 SysName = openflow:2
10	34.999090812	96:93:fb:f1:77:f9	CayeeCom_00…	LLDP	85	TTL = 4919 SysName = openflow:2
11	37.106490534	10.0.0.1	10.0.0.2	RTP	1370	PT=MPEG-II transport streams, SSRC=0x22DF71CF, Seq=4844, Time=37326460
12	38.079786732	10.0.0.1	10.0.0.2	RTP	1370	PT=MPEG-II transport streams, SSRC=0x22DF71CF, Seq=4845, Time=42614151
13	38.079838585	10.0.0.1	10.0.0.2	RTP	1370	PT=MPEG-II transport streams, SSRC=0x22DF71CF, Seq=4846, Time=42575190
14	38.079864648	10.0.0.1	10.0.0.2	RTP	1370	PT=MPEG-II transport streams, SSRC=0x22DF71CF, Seq=4847, Time=42578952
15	38.079889876	10.0.0.1	10.0.0.2	RTP	1370	PT=MPEG-II transport streams, SSRC=0x22DF71CF, Seq=4848, Time=42582713
16	38.079914328	10.0.0.1	10.0.0.2	RTP	1370	PT=MPEG-II transport streams, SSRC=0x22DF71CF, Seq=4849, Time=42586475
17	38.079938270	10.0.0.1	10.0.0.2	RTP	1370	PT=MPEG-II transport streams, SSRC=0x22DF71CF, Seq=4850, Time=42590236
18	38.079963324	10.0.0.1	10.0.0.2	RTP	1370	PT=MPEG-II transport streams, SSRC=0x22DF71CF, Seq=4851, Time=42593998

Ilustración 28 Captura en Wireshark de la topología SDN en GNS3.
Autores: Los investigadores.

En la imagen anterior se puede apreciar las muestras de los diferentes paquetes capturados con el sniffer entre la pc1 (10.0.0.1) y la pc2 (10.0.0.2), el protocolo que más está presente es el RTP debido a que es el responsable de la transmisión de vídeo, este protocolo cuenta con una longitud de 1370 Bytes.

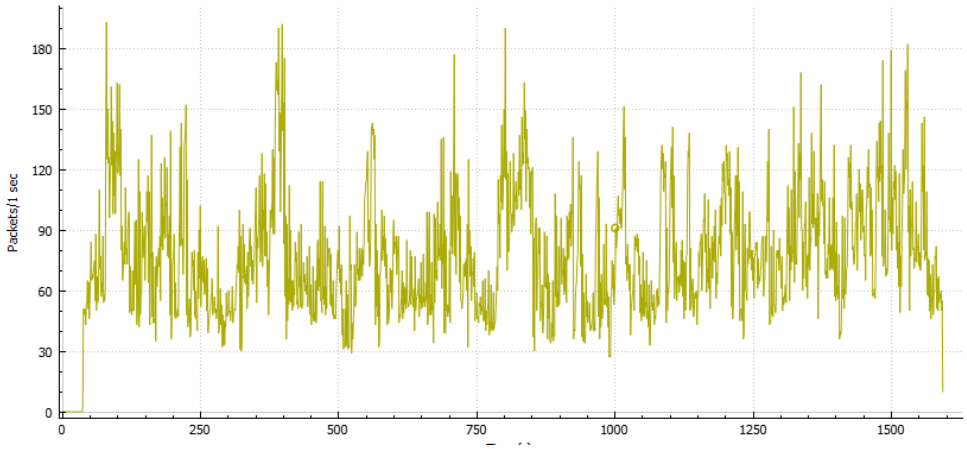


Ilustración 29 Gráfica estadística de la transmisión del vídeo en HD.
Autores: Los investigadores.

En la ilustración 37 se encuentra la gráfica estadística de la transmisión del vídeo en HD en relación del tiempo y la cantidad de paquetes transmitidos por segundo.

Topología de red con arquitectura TCP/IP.

Esta topología es más sencilla que la mencionada en el punto anterior ya que cuenta con dos switch ethernet de capa dos enlazados a los diferentes dispositivos virtualizados los cuales uno hará de servidor streaming y el resto de clientes

de este servicio. Y para las capturas de tráfico se utilizó Wireshark (Ilustración 38).

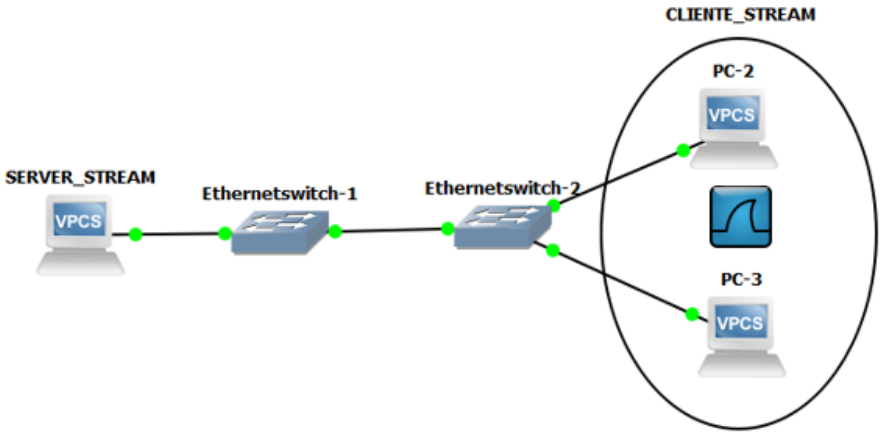


Ilustración 30 Topología de red con arquitectura TCP/IP en GNS3.
Autores: Los investigadores.

En esta topología de igual manera se realizó la captura de tráfico durante las transmisiones de vídeo a los host clientes como se puede ver a continuación en la ilustración 39.

	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.0.1	10.0.0.2	RTP	1370	PT=MPEG-II transport streams, SSRC=0x3319643F, Seq=15961, Time=714240404
2	0.003471061	10.0.0.1	10.0.0.2	RTP	1370	PT=MPEG-II transport streams, SSRC=0x3319643F, Seq=15962, Time=714200251
3	0.003515480	10.0.0.1	10.0.0.2	RTP	1370	PT=MPEG-II transport streams, SSRC=0x3319643F, Seq=15963, Time=714205501
4	0.003541943	10.0.0.1	10.0.0.2	RTP	1370	PT=MPEG-II transport streams, SSRC=0x3319643F, Seq=15964, Time=714210751
5	0.003566766	10.0.0.1	10.0.0.2	RTP	1370	PT=MPEG-II transport streams, SSRC=0x3319643F, Seq=15965, Time=714216001
6	0.003591470	10.0.0.1	10.0.0.2	RTP	1370	PT=MPEG-II transport streams, SSRC=0x3319643F, Seq=15966, Time=714219938
7	0.003616472	10.0.0.1	10.0.0.2	RTP	1370	PT=MPEG-II transport streams, SSRC=0x3319643F, Seq=15967, Time=714223876
8	0.003641053	10.0.0.1	10.0.0.2	RTP	1370	PT=MPEG-II transport streams, SSRC=0x3319643F, Seq=15968, Time=714227813
9	0.154613398	10.0.0.1	10.0.0.2	RTP	1370	PT=MPEG-II transport streams, SSRC=0x3319643F, Seq=15969, Time=714231751
10	0.154777523	10.0.0.1	10.0.0.2	RTP	1370	PT=MPEG-II transport streams, SSRC=0x3319643F, Seq=15970, Time=714200251
11	0.154871831	10.0.0.1	10.0.0.2	RTP	1370	PT=MPEG-II transport streams, SSRC=0x3319643F, Seq=15971, Time=714205501
12	0.154936581	10.0.0.1	10.0.0.2	RTP	1370	PT=MPEG-II transport streams, SSRC=0x3319643F, Seq=15972, Time=714210751
13	0.154999913	10.0.0.1	10.0.0.2	RTP	1370	PT=MPEG-II transport streams, SSRC=0x3319643F, Seq=15973, Time=714216001
14	0.155090237	10.0.0.1	10.0.0.2	RTP	1370	PT=MPEG-II transport streams, SSRC=0x3319643F, Seq=15974, Time=714219938
15	0.155156736	10.0.0.1	10.0.0.2	RTP	1370	PT=MPEG-II transport streams, SSRC=0x3319643F, Seq=15975, Time=714223876
16	0.155221300	10.0.0.1	10.0.0.2	RTP	1370	PT=MPEG-II transport streams, SSRC=0x3319643F, Seq=15976, Time=714227813
17	0.206178684	10.0.0.1	10.0.0.2	RTP	1370	PT=MPEG-II transport streams, SSRC=0x3319643F, Seq=15977, Time=714231751
18	0.206221215	10.0.0.1	10.0.0.2	RTP	1370	PT=MPEG-II transport streams, SSRC=0x3319643F, Seq=15978, Time=714200251
19	0.206245744	10.0.0.1	10.0.0.2	RTP	1370	PT=MPEG-II transport streams, SSRC=0x3319643F, Seq=15979, Time=714205501
20	0.206269558	10.0.0.1	10.0.0.2	RTP	1370	PT=MPEG-II transport streams, SSRC=0x3319643F, Seq=15980, Time=714210751
21	0.206292476	10.0.0.1	10.0.0.2	RTP	1370	PT=MPEG-II transport streams, SSRC=0x3319643F, Seq=15981, Time=714216001
22	0.206315309	10.0.0.1	10.0.0.2	RTP	1370	PT=MPEG-II transport streams, SSRC=0x3319643F, Seq=15982, Time=714219938

Ilustración 31 Captura en Wireshark de la topología TCP/IP en GNS3.
Autores: Los investigadores.

Cabe destacar en este caso la en las muestras de tráfico solo se ve paquetes RTP capturados por motivo de que en TCP/IP no presentó un tráfico antes de la transmisión del vídeo.

Cuarto Escenario.

Descripción del escenario Cuatro.

En este escenario se implementaron tres topologías tipo árbol, una híbrida, una con SDN y otra con arquitectura TCP/IP. En las topologías se midió el ancho de banda, latencia mínima, latencia media, latencia máxima y jitter. Para medir el ancho de banda se utilizó la aplicación Jperf y obtener el valor de las latencias y jitter se efectuaron pruebas de envío de 1000 y 2000 paquetes. Cada una de estas pruebas se repitió cincuenta veces, dándonos un total de trescientas pruebas realizadas en este escenario con las tres topologías.

4.1.1.1. Escenario cuatro con arquitectura Híbrida.

Se utilizó una topología híbrida que está compuesta por arquitecturas SDN y TCP/IP (Ilustración 40). Esta red integra los protocolos utilizados en las redes TCP/IP como los usados en SDN. La topología está compuesta por siete switches que para SDN que admiten el protocolo OpenFlow y ocho switches utilizados para las redes TCP/IP. También cuenta con diez ordenadores, dos de ellas conectadas a un switch OpenFlow y ocho enlazadas a switches para la red TCP/IP. Se integra un controlador remoto que es OpenDayLight y este solo está enlazado a los switches OpenFlow.

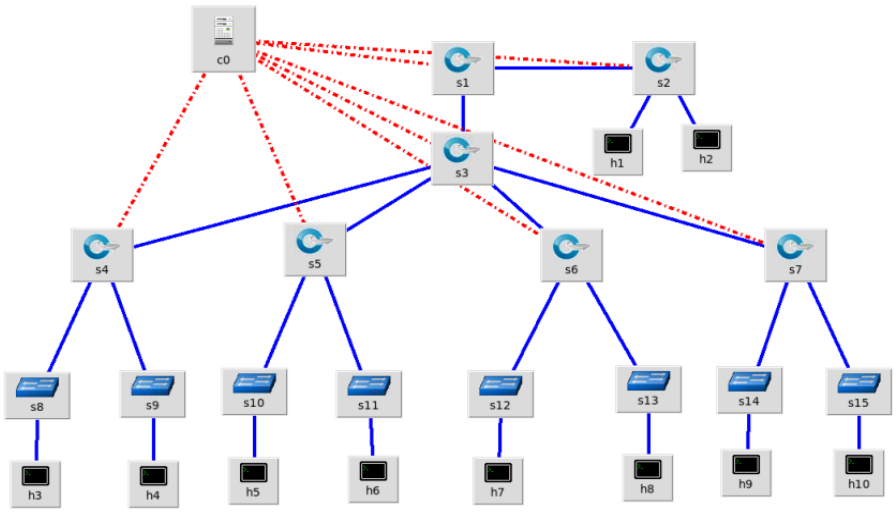


Ilustración 10 Topología de red Híbrida del cuarto escenario.
Autores: Los investigadores.

Los resultados de las pruebas de latencias y jitter se muestran en la tabla 20. Los valores registrados en la tabla corresponden a las medias de las cincuenta pruebas realizadas tanto para 1000 paquetes como para 2000 paquetes.

N° paquetes enviados en 50 repeticiones	Latencia min	Latencia media	Latencia máx.	Jitter
1000	0.418	0.841	10.072	0.808
2000	0.414	0.892	10.312	0.977

Tabla 1 Resultados de medias de latencia y jitter de la arquitectura Híbrida en el escenario cuatro.
Autores: Los investigadores.

Escenario cuatro con arquitectura SDN.

Esta topología es similar a la híbrida, solo que aquí se utilizaron quince switches destinados específicamente para SDN. Todos los switches se conectaron directamente al controlador que fue programado para su uso de forma remota. Además, también se integró diez ordenadores

conectados a los switches terminales. La topología se observa en la ilustración 41.

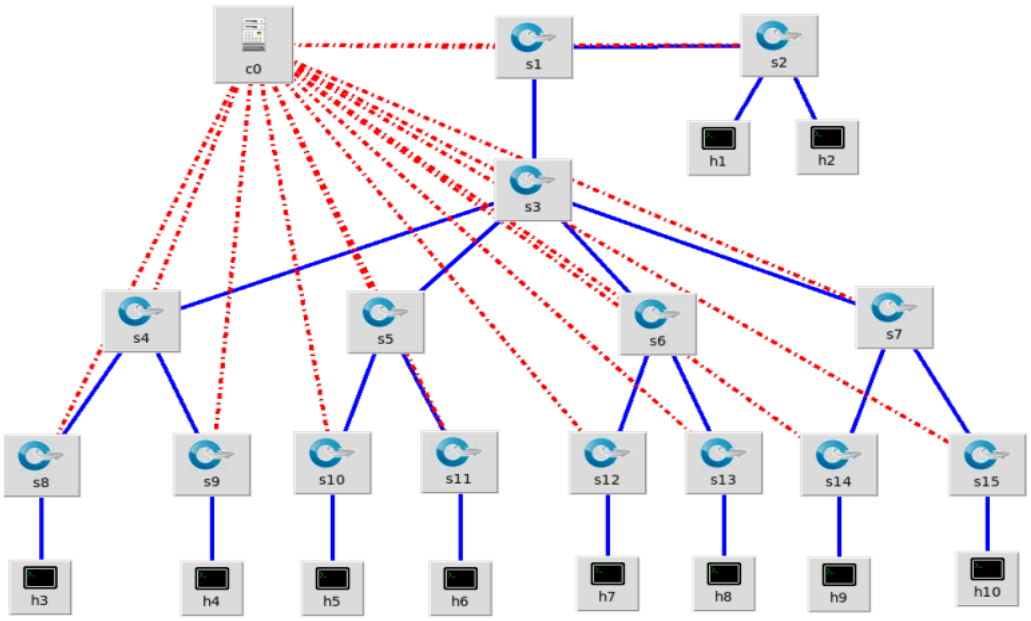


Ilustración 11 Topología de red SDN del cuarto escenario.
Autores: Los investigadores.

Las pruebas se realizaron enviando paquetes solamente entre el host h3 y el host h10 que estaban situados en los extremos de la red. En la tabla 21 se reflejan los resultados en promedio de las pruebas efectuadas en esta topología.

N° paquetes enviados en 50 repeticiones	Latencia min	Latencia media	Latencia máx.	Jitter
1000	0.422	0.868	8.175	0.967
2000	0.420	0.936	6.217	0.754

Tabla 2 Resultados de medias de latencia y jitter de la arquitectura SDN en el escenario cuatro.
Autores: Los investigadores.

Escenario cuatro con arquitectura TCP/IP.

Para este escenario se realizó una topología en árbol basándose a las realizadas en la red híbrida y con SDN. Para este caso en particular, se usaron quince switches que

admiten solo los protocolos para las redes con arquitectura TCP/IP. También la misma cantidad de host conectados a los terminales, pero no se contó con la presencia de un controlador para monitorizar la red, debido a que solo son dispositivos característicos de SDN (Ilustración 42).

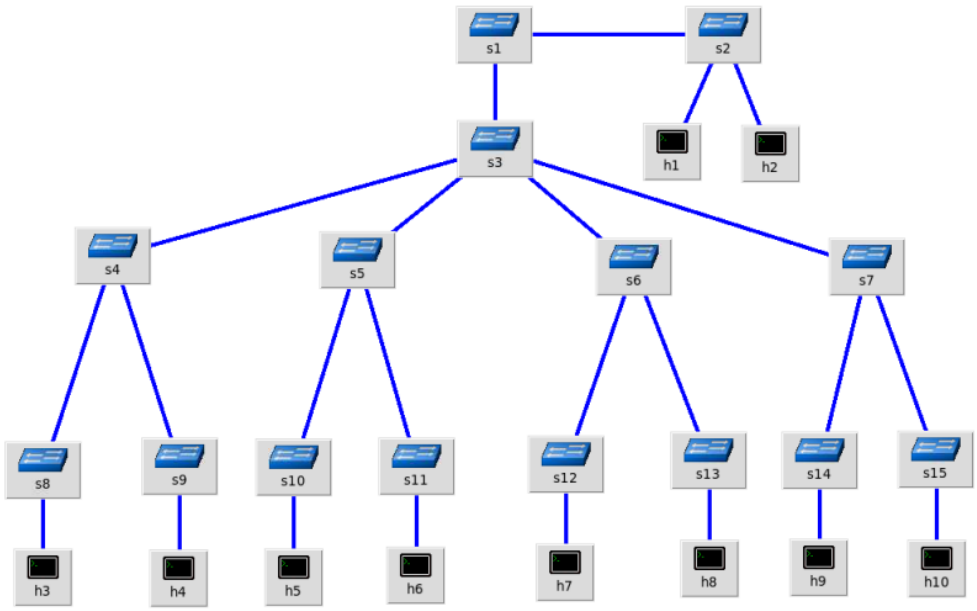


Ilustración 12 Topología de red TCP/IP del cuarto escenario.
Autores: Investigadores.

A continuación, en la tabla 22 se muestran los promedios de los valores resultantes de las pruebas con 1000 y 2000 paquetes enviados desde el host h3 hasta el host h10.

N° paquetes enviados en 50 repeticiones	Latencia min	Latencia media	Latencia máx.	Jitter
1000	0.439	0.725	12.443	0.791
2000	0.422	0.704	15.646	0.749

Tabla 3 Resultados de medias de latencia y jitter de la arquitectura TCP/IP en el escenario cuatro.
Autores: Investigadores

Comparación estadística de resultados del escenario Cuatro.

En esta sección se comparó las tres topologías descritas anteriormente que son la Híbrida, SDN y TCP/IP. Primero se obtuvo la media o el promedio de las latencias, esto se puede visualizar en la tabla 23.

Estadísticas de medias de Latencia					
	Tipos de redes	N	Media	Desviación estándar	Media de error estándar
Latencia mínima	Híbrida	2	0,41600	0,002828	0,002000
	SDN	2	0,42100	0,001414	0,001000
	TCP/IP	2	0,43050	0,012021	0,008500
Latencia media	Híbrida	2	,86650	0,036062	0,025500
	SDN	2	,90200	0,048083	0,034000
	TCP/IP	2	,71450	0,014849	0,010500
Latencia máxima	Híbrida	2	7,52700	0,008485	0,006000
	SDN	2	7,19600	1,384515	0,979000
	TCP/IP	2	14,04450	2,264863	1,601500

Tabla 4 Comparación estadísticas de medias de Latencia del escenario cuatro. Autores: Investigadores.

En la ilustración 43 se puede observar una gráfica que representa los promedios de latencias que resultaron de la topología híbrida, SDN y TCP/IP.

Para comparar las medias de latencias no se utilizó el método estadístico "T" como en el escenario dos; sino, que se implementó un método conocido por su capacidad de comparar tres o más variables denominado ANOVA. Se utilizó el software estadístico R para realizar un código que implemente el método ANOVA.

Los resultados obtenidos en la comparación de latencias son los que se muestran en la tabla 24, allí se encuentran las

métricas que se compararon y el valor del estadístico **P**. La prueba ANOVA al igual que la T Students se basa en escoger la hipótesis nula o la alternativa dependiendo el valor que tome el estadístico P. Si el valor de P es menor a 0,5 se concluye que existe una diferencia significativa entre las variables que se comparan, y si es mayor no existe diferencia significativa.

También se logró determinar el valor de las diferencias de las latencias, estas se encuentran en la tabla 25. Los valores que se encuentran en negrita son los que corresponden a las diferencias significativas que ocurrieron en la comparación estadística.

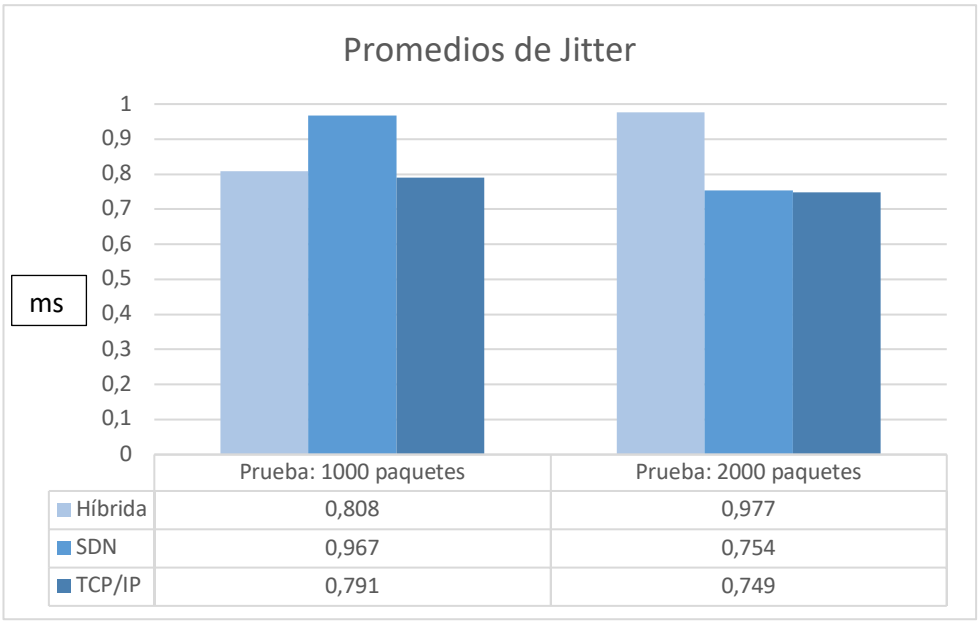
Latencias	**Tipo de red**	**Diferencia de medias**
Latencia mínima	SDN-Híbrida	0.0050
	TCP/IP-Híbrida	0.0145
	TCP/IP-SDN	0.0095
Latencia media	SDN-Híbrida	0.0355
	TCP/IP-Híbrida	**-0.1520**
	TCP/IP-SDN	**-0.1875**
Latencia máxima	SDN-Híbrida	-0.3310
	TCP/IP-Híbrida	**6.5175**
	TCP/IP-SDN	**6.8485**

Con el Jitter se efectuó el mismo procedimiento, primero se obtuvo los resultados de las medias de las tres topologías, lo cual se presenta en la tabla 26.

Estadísticas de medias de Jitter					
Jitter	Tipos de redes	N	Media	Desviación estándar	Media de error estándar
	Híbrida	2	0,89250	0,11950	0,08450
	SDN	2	0,86050	,015061	0,10650
	TCP/IP	2	0,77000	0,02969	0,02100

Tabla 5 Comparación estadísticas de medias de jitter del escenario cuatro.
Autores: Los investigadores.

En la gráfica 44 que se encuentra en la ilustración se aprecia los promedios de Jitter de las topologías utilizadas en este escenario. Los promedios mostrados corresponden a las pruebas de 1000 y 2000 paquetes realizadas cincuenta veces cada una.



Para comparar los valores de Jitter de las tres topologías también se utilizó la prueba ANOVA y observando el resultado del estadístico P se concluye que no existe diferencia significativa de valores entre las tres topologías.

Una de las ventajas de utilizar la prueba ANOVA es que se pudo utilizar una herramienta para conocer la diferencia de medias.

Etapa tres: Análisis de tráfico del Tercer Escenario.

La evaluación de diversos escenarios con aplicaciones de uso común es un punto importante para determinar qué tan eficiente es una red y cómo se comporta en un entorno donde ya se podría involucrar al usuario final. Esta etapa se enfoca en el análisis de rendimiento a través de capturas de tráfico durante la transmisión de vídeo en directo planteada en el escenario 3 de esta investigación.

Determinación de umbral de transmisión óptima.

Para la determinación del umbral de transmisión en las calidades establecidas se realizó a través del cálculo de la tasa de bits por segundo en relación del tamaño de los paquetes transmitidos con un tamaño de 1370 Bytes. A continuación, se puede visualizar en las ilustraciones 45 y 46 el cálculo para el vídeo en HD y FHD:

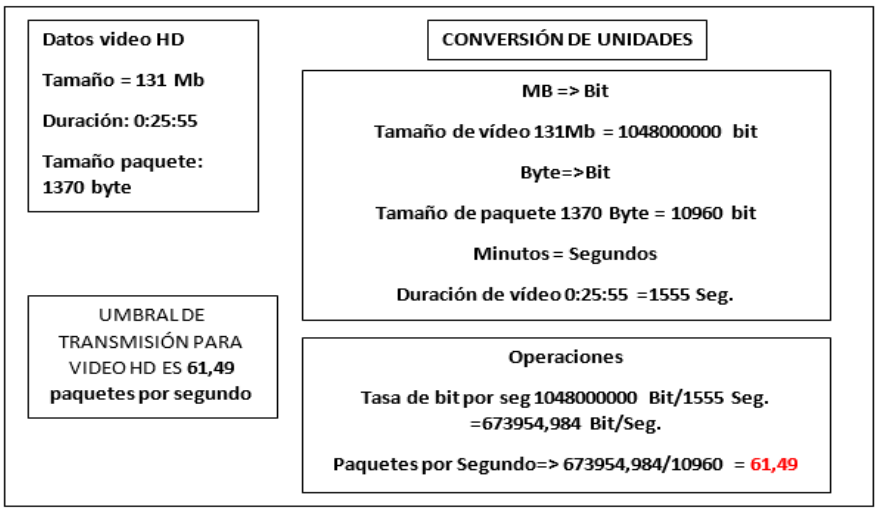
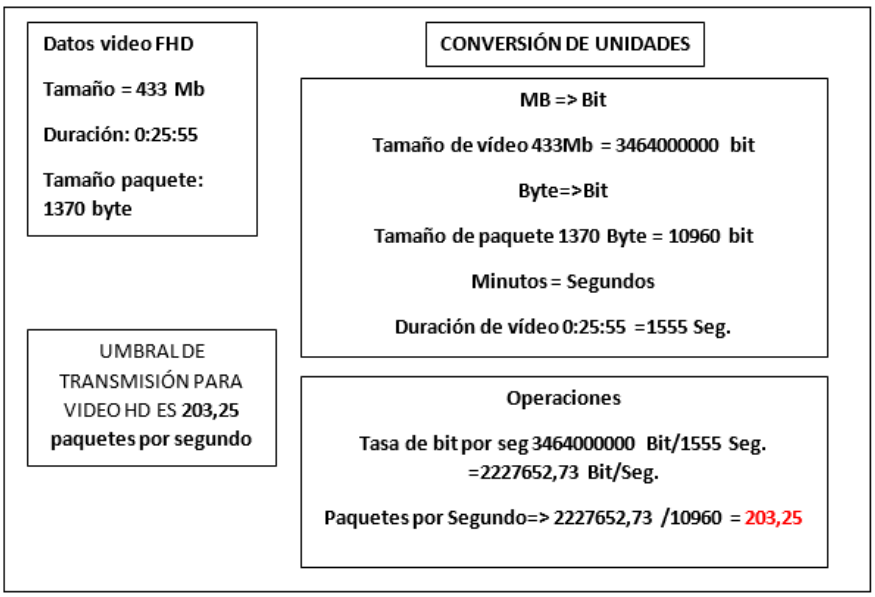


Ilustración 13 Determinación de umbral de transmisión óptima en video HD.
Autores: Los investigadores.



Como resultado de estos cálculos para el vídeo en HD se obtuvo que el umbral necesario para mantener una buena transmisión es 61,49 paquetes por segundo (pps) y para el vídeo en FHD en 203,24 pps.

Protocolos involucrados durante la transmisión.

Durante las pruebas realizadas se obtuvo varias muestras de diferentes paquetes que hacen referencia a los protocolos necesarios para la trasmisión de la información. En ambas topologías nos encontramos que se ocuparon los mismos protocolos, pero en la red con SDN al contar con el controlador ODL, se presenta un protocolo adicional enfocado al descubrimiento de los dispositivos de la red.

Análisis de captura de tráfico con Wireshark.

Wireshark permitió la captura los diferentes tipos de paquetes que se enviaron durante las transmisiones de streaming en las pruebas realizadas. En este apartado se analizará a detalle cada una de estas capturas en función de las estadísticas obtenidas con el sniffer y el umbral obtenido en puntos anteriores.

Análisis en función de umbral de transmisión.

Este análisis se lo realiza enfocándose en los paquetes RTP o UDP, encargados de segmentar la información para su transporte entre los dispositivos de la red.

- **Video HD**

  Con la gráfica 48 sobre el envío de paquetes se puede visualizar que ambas topologías llegan a superar el umbral de transmisión incluso teniendo patrones similares en función del envío de paquetes. Lo que en los hosts clientes no se perdió la calidad durante la transmisión no se presentaron retardos que dañen la calidad del vídeo o algún corte de audio.

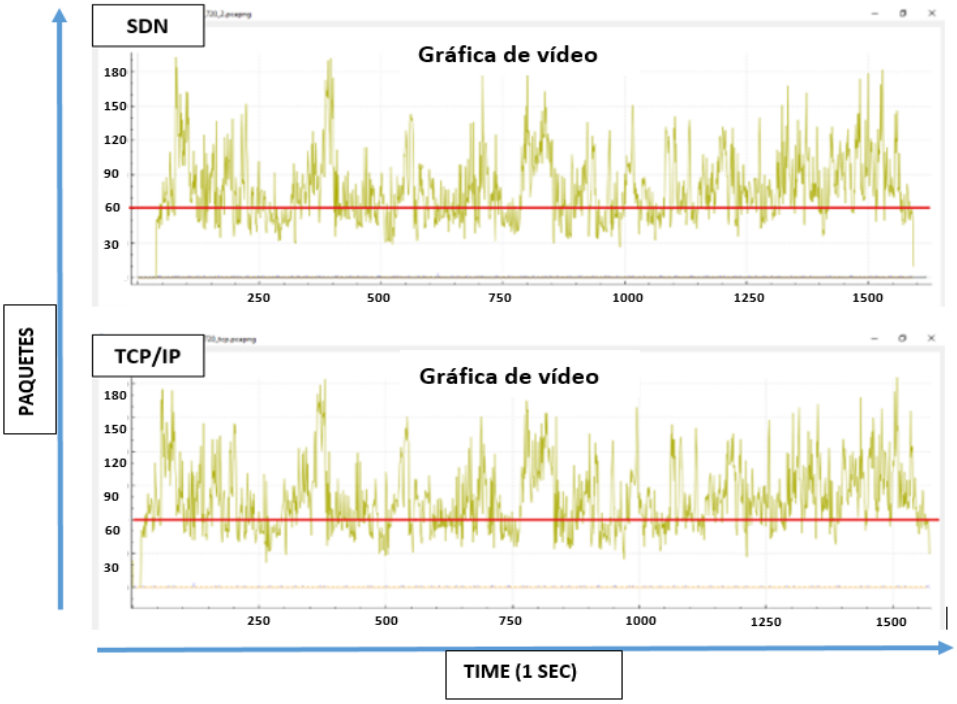


Ilustración 14 Análisis en función de umbral de transmisión del video HD.
Autores: Investigadores.

Entre otras estadísticas obtenidas en esta transmisión se puede apreciar que en amabas redes se obtiene valores similares en la tabla 29, pero de cierta forma en SDN muestra valores ligeramente más bajos que su contraparte.

- **Video FHD.**

  Con esta calidad de vídeo se obtuvo gráficas de envío de paquetes por segundo similares que en la trasmisión en SDN, pero como valores algo más elevados.

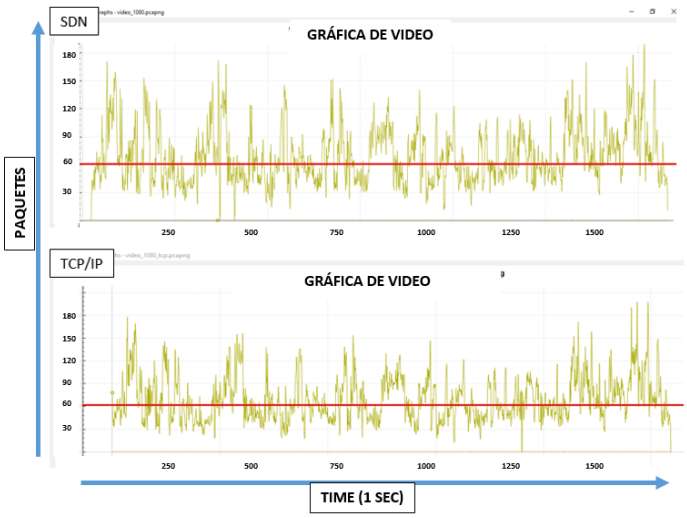


Ilustración 15 Análisis en función de umbral de transmisión del video FHD.
Autores: Los investigadores.

En cuestión de las características adicionales se pudo apreciar que en cuestión del consumo del ancho de banda en la red SDN se mantuvo más bajo con una diferencia aproximada de 500 kb/seg. Pero en cuestión de la media de paquetes por segundo se mantuvo en valores superiores al umbral de transmisión a diferencia de TCP/IP que con esta calidad de vídeo prácticamente se mantuvo debajo del umbral (Tabla 30).

Análisis en función de los protocolos.

En este punto se analizó el tráfico obtenido en cada uno de los protocolos partícipes durante la transmisión streaming.

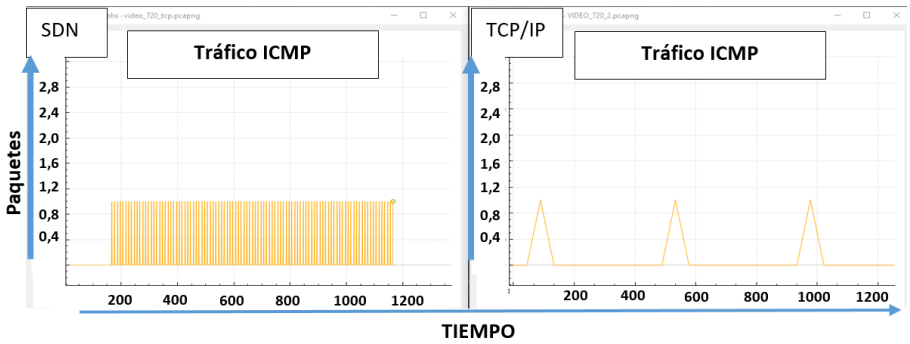
- **ARP.**

  En este protocolo se puede apreciar un cambio en las dos arquitecturas, aquí se puede ver que el protocolo tiene una mayor repercusión en la red TCP/IP ya que es quien le permite descubrir y determinar las mac de origen y destino durante la transmisión (Ilustración 50). Lo que no sucede en SDN donde

eso lo realiza con otro protocolo enfocado al descubrimiento de la red.

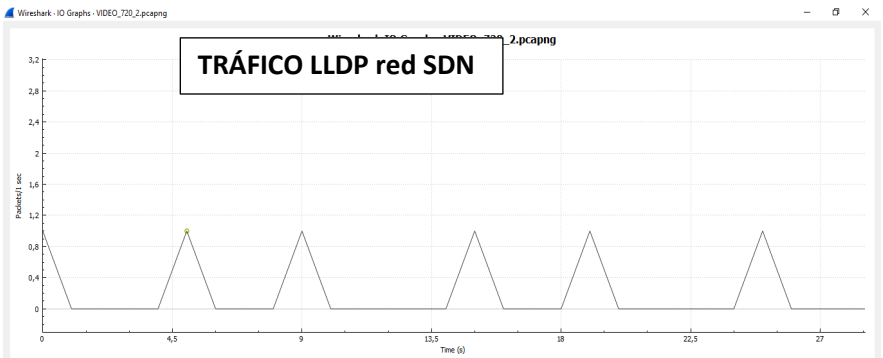
- **ICMP.**

  Estas estadísticas representan los mensajes de error que se dieron durante la transmisión, en TCP/IP este tráfico es más constante prácticamente secuencial a diferencia de SDN el cual es un tráfico menos intenso (Ilustración 51).



- **LLDP.**

  La muestra obtenida se dio solo en la red con arquitectura SDN. Este tráfico representa a la comunicación entre los dispositivos y el controlador ODL, como detalle de la muestra es que es constante antes y durante la transmisión. Este paquete cuenta con 85 bytes de longitud (Ilustración 52).



Con este proyecto de investigación se logró medir el desempeño tanto de redes con arquitectura SDN como TCP/IP, utilizando métricas de evaluación y analizando el

tráfico generado en el escenario tres de la segunda etapa. Con respecto a los resultados obtenidos en las diferentes etapas se pudo considerar los siguientes puntos:

- **Etapa de identificación de controladores, herramientas de emulación y protocolos que pueden implementarse en redes definidas por software.**

  En la primera parte de esta etapa se identificaron los protocolos que interactúan en las redes definidas por software y se tomó en cuenta principalmente la función que cumplen dentro de la red. Algunos de ellos están enfocados a la parte de gestión, configuración de dispositivos y de transporte en la capa de red, pero de todos estos protocolos sobresalió uno que es el más importante y popular en SDN que es OpenFlow. El protocolo OpenFlow como se ha escrito en etapas anteriores se encarga principalmente de permitir la comunicación de los planos que conforman la arquitectura de una red que se define por software. Incluso se llega a asumir que si no existiera el protocolo OpenFlow las redes SDN no se lograrían comunicar correctamente con los dispositivos, debido a que constituye gran parte de las funciones que se deben cumplir en estas redes como crear tablas de flujo y administrarlas. El protocolo logra realizar la administración de tablas de flujos porque cuenta con una interfaz web denominada OpenFlow Manager.

  En esta etapa también se identificaron los controladores más populares para SDN entre ellos están NOX, POX, Beacon, Floodlight, OpenDayLight y Ryu. En la selección del controlador para este proyecto se descartó NOX, POX y Beacon porque no admiten el uso de versiones actuales del protocolo OpenFlow. La decisión estaba entre Floodlight, OpenDayLight y Ryu, los tres presentaban ventajas como que

admiten la última versión del protocolo OpenFlow, cuentan con interfaz web, pero solo Floodlight y OpenDayLight son multi-plataformas.

Entre los controladores se eligió OpenDayLight debido a que su interfaz web era la más amigable y presentaba opciones más completas destinadas a la administración de dispositivos, pero este controlador lanza nuevas versiones en muy poco tiempo y esto causó un inconveniente en el proyecto debido a que algunas opciones quedaron obsoletas y no se podía realizar una administración de la red utilizando APIs.

Se seleccionó dos emuladores para redes SDN uno que es Mininet que surgió junto con las redes SDN y permite realizar diferentes tipos de topologías mediante su línea de comandos o programando un script en lenguaje Python, además cuenta con una interfaz gráfica en la que se pueden visualizar las topologías y editar algunos parámetros. El otro emulador elegido fue GNS3, que, si bien cierto no está dedicado para uso exclusivo de SDN, pero en la actualidad los proyectos de SDN se están enfocando hacia esta plataforma debido a que permite emular dispositivos que existen en el mundo real tan solo cargando las ISO de su sistema.

- **Etapa de diseño e implementación de escenarios prácticos.**

  En esta etapa se diseñaron e implementaron cuatro escenarios en los que se realizaron pruebas para comprobar el rendimiento de las redes con arquitectura SDN y TCP/IP.

  El primer escenario se enfocó a medir el ancho de banda que alcanzaría cada una de estas topologías basadas en TCP/IP y SDN. Entre los resultados que se presentaron luego

de generar tráfico UDP en toda la red a través de la herramienta jperf se determinó que la red SDN admite un mayor ancho de banda ya que su umbral se estableció en 112 Mbytes y por su parte la red con arquitectura TCP/IP presentó su umbral en 60 Mbytes. Esta diferencia de ancho de banda que admite cada red permite visualizar que red se comportaría mejor ante una cantidad considerable de tráfico.

En el segundo escenario en las pruebas comparativas que se realizaron en el software estadístico R se pudo llegar a conocer que existe diferencia entre las latencias mínima, media y máxima, debido a que la red SDN está optimizada por el controlador. En este escenario también se midió el Jitter en el cual los resultados arrojaron que no existe diferencia, pero comparando los valores de las medias se puede verificar que la arquitectura TCP/IP tuvo menor tiempo de la variación de paquetes.

El cuarto escenario fue similar al segundo escenario, solo que en este se compararon tres topologías una híbrida, una SDN y una TCP/IP. Los resultados se obtuvieron mediante una prueba estadística para tres variables. En la evaluación de latencias la red que se desempeñó mejor fue SDN, la red híbrida tuvo mayor latencia porque en ella intervenían protocolos tanto de SDN como de TCP/IP, esto ocasionó un ligero retraso en el envío de paquetes. La red TCP/IP a pesar de no tener el mejor resultado en latencia en Jitter tiene una pequeña mejoría comparada con las otras dos redes, aunque estadísticamente resulte que no exista ninguna diferencia.

- **Etapa de Análisis de tráfico.**

  Aquí se tomó como referencia al escenario tres debido a que está enfocado a capturas de tráfico a través de la herramienta Wireshark. Este análisis se lo realizó por separado cada una de las transmisiones donde en la primera transmisión se la hizo con un vídeo en HD el cual en ambas redes las estadísticas dieron valores similares en paquetes como RTP el cual se encargaba de la transmisión. En paquetes ARP y ICMP se pudo aprecia que en la red con TCP/IP presentó una mayor interacción con estos protocolos a diferencia de SDN que tuvo muy poco tráfico generado. Pero SDN presentó un nuevo paquete LLDP el cual se encargaba de mantener informado al controlador de algún inconveniente durante la transmisión, integrando de cierta forma las funcionalidades los paquetes ARP e ICMP.

  En el caso del vídeo en FHD se presentaron cambios notables entre ambas arquitecturas, aquí se pudo apreciar como en el caso de SDN se mantuvo sobre el umbral de transmisión obtenido lo que garantizó una transmisión estable en los equipos clientes. En cado de TCP/IP su rendimiento en los primeros cinco min de transmisión fue similar a su contraparte, pero a medida que pasada el tiempo de transmisión se empezaron a notar retardos en la red con lo que se perdió en ciertos puntos calidad tanto en el vídeo como en el audio.

  Es importante realizar un estudio minucioso de las herramientas necesarias para trabajar tanto con redes SDN como con redes TCP/IP, con el fin de elegir la que mejor se adapte a las necesidades de cada investigación. También para obtener datos que reflejen la diferencia en el desempeño de estas dos redes se deben realizar escenarios

evaluando distintas métricas y analizar su tráfico para reconocer su comportamiento.

A continuación, se presentan tres conclusiones específicas de esta investigación.

- La identificación de protocolos, controladores y emuladores ayudó a establecer con qué herramientas de software se puede contar y cuáles de estas se acoplan a los requerimientos de esta investigación. La selección del controlador, en este caso OpenDayLight y el protocolo OpenFlow es de gran importancia debido a que permite tener una correcta interacción entre los dispositivos y la administración de las redes. Así mismo esta investigación se enfatizó en la selección de dos emuladores que son Mininet y GNS3 que brindan las prestaciones correctas para permitir crear diferentes tipos de topologías y realizar evaluaciones.
- En la etapa dos de la investigación se realizó pruebas en cuatro escenarios donde se evaluó el desempeño de ambas arquitecturas. La conclusión de cada arquitectura se presenta a continuación:

✓ En el primer escenario, luego de las respectivas evaluaciones, se pudo obtener que SDN ante una mayor intensidad de tráfico llegan a obtener casi el doble de capacidad que TCP/IP al poder manejar tráfico UDP.

✓ En el segundo escenario al evaluar las latencias mínimas, medias y máximas en redes SDN y TCP/IP se concluye que existe una diferencia significativa; es decir, que no son iguales las medias y la red que presentó menor tiempo de latencias fue SDN. Pero en el caso del jitter, estadísticamente las medias resultaron iguales en ambas redes.

✓ En el escenario tres, luego de las pruebas capturando el tráfico, se puede apreciar que los paquetes tanto en una red

como en otra son diferentes en su intensidad, en SDN no se da prioridad a paquetes como ARP, ICMP por que por su parte actúa el paquete LLCP. Por otra parte, con TCP/IP sus prioridades después del paquete RTP son los paquetes ARP e ICMP porque son los paquetes que le permiten verificar el estado de la red y sus dispositivos internamente.

- ✓ Se concluye en el cuarto escenario que la topología que tuvo mejor desempeño en los valores de latencia fue la que tuvo arquitectura SDN, seguida de la Híbrida y la TCP. La estadísticamente en Jitter las tres topologías no presentó diferencias abismales por lo que se concluye que el desempeño es igual.

- En los resultados análisis de tráfico en las pruebas realizadas con respecto al escenario tres, se obtuvo datos de jitter, ancho de banda y la media de paquetes a través de estadísticas de Wireshark las cuales en cierta forma mostraron mejoras en ciertos aspectos en ambas redes. En cuestión de consumo de ancho de banda en SDN fue menor lo que permite ver que SDN optimiza de mejor manera la red para que tenga un menor consumo. En cuestión de paquetes por segundo en SDN cuando se transmitió una mayor cantidad de datos se mantuvo por encima del umbral de transmisión lo cual se comprobó en el vídeo, que no tuvo retardos o pérdidas de calidad a diferencia de TCP/IP donde si hubo a medida que avanzaba la transmisión.

De acuerdo a las conclusiones se recomienda lo siguiente:

- Se recomienda revisar suficiente documentación tanto de fabricantes como de proyectos de investigación con la finalidad de obtener un panorama claro sobre la actualidad de las redes definidas por software. Estas redes se encuentran en constante evolución y los principales desarrolladores tecnológicos están apoyando continuamente a estos

proyectos debido al impacto que causan en el sector de las telecomunicaciones.

- Es recomendable que al evaluar el desempeño de diferentes tipos de arquitecturas de redes se diseñen e implementen escenarios de prueba. Con la finalidad de poder analizar el desempeño que tienen las redes con respecto a métricas de evaluación.
- Para un mayor análisis del tráfico es recomendable poner a prueba ambas arquitecturas con otro tipo de archivos para apreciar como gestiona el envío de paquetes desde un punto a otro. Incluso enviando archivos en ambas direcciones para evaluar su rendimiento tanto en métricas de latencia, jitter y ancho de banda.

BIBLIOGRAFÍA

[1] B. Underdahl and G. Kinghorn, Software Defined Networking for Dymmies, Cisco Spec. Ner Jersey: Jhon Wiley & Sons, Inc., 2015.

[2] R. J. S. A, "Redes de distribución de contenidos basadas en redes definidas por software," 2016.

[3] C. Gonzalez, O. Flauzac, and F. Nolot, "Evolución y Contribución para el Internet de las Cosas por las emergentes Redes Definidas por Software," Memorias Congr. UTP, vol. 1, no. 1, pp. 28–33, 2018.

[4] D. Bolatti, R. Calcagno, C. Cuevas, S. Gramajo, and R. Scappini, "Modelo para la evaluación de performance mediante identificación de tráfico y atributos críticos en Redes Definidas por Software," 2015.

[5] H. C. E. Carlos Lester Dueñas Santos,Yanko Antonio Marín Muro, "(PDF) Analysis of the performance of an SDN network over a traditional network in LAN environments," ResearchGate, 2017.

[6] B. D. X. Albán Pablo Fernando, "Diseño e Implementación del prototipo de una red definida por software (SDN) en la universidad de las Fuerzas Armadas ESPE."

[7] J. Escobar, "Control de flujo de una red definida por software usando sensores térmicos," 2015.

[8] Cordero Christian Fabián, "Diseño y Despliegue de Funciones de Red Virtualizadas (NFV) usando Redes Definidas

por Software (SDN) dentro de una infraestructura Virtual, aplicando balanceo de carga y seguridad distribuida en IPv6," 2017.

[9] W. Stallings, "Comunicaciones y Redes de Computadores," p. 896, 2004.

[10] CISCO, "Cisco Visual Networking Index: Forecast and Trends, 2017–2022 White Paper - Cisco," 2019.

[11] W. I. Intriago Romero, "Estudio del protocolo Openflow usando el modelo de red definida por Software (Software Define Networks). Caso de estudio la Universidad Técnica de Manabí," 2017.

[12] S.-Y. Wang and J. C. Wen, "EstiNet X | EstiNet - Simulador," 2015. [Online]. Available: http://www.estinet.com/ns/?page_id=21140. [Accessed: 17-Jul-2019].

[13] R. D. Nasution, "Red definida por software (SDN) en base a una infraestructura de software de libre distribución," vol. 3, no. 2, pp. 54–67, 2015.

[14] J. F. KUROSER and K. W. ROSS, Redes de computadoras: Un enfoque descendente, Quinta. Madrid, 2010.

[15] A. S. Tanenbaum and D. J. Wetherall, Redes de Computadoras, 5ta Edición, Quinta. Washington, 2012.

[16] H. Solís and L. Martínez, "Características de una arquitectura de red." [Online]. Available: http://cidecame.uaeh.edu.mx/lcc/mapa/PROYECTO/libro35/12_caractersticas_de_la_arquitectura_de_red.html.

[17] Redhat, "virtualization," 1, 2019. [Online]. Available:

https://www.redhat.com/es/topics/virtualization.

[18] ONF, "Definición de redes definidas por software (SDN) - Open Networking Foundation," 2013. [Online]. Available: https://www.opennetworking.org/sdn-definition/?nab=0. [Accessed: 19-Jun-2019].

[19] B. Beach and B. Beach, "AWS Architecture Overview," Pro Powershell Amaz. Web Serv., no. 1, pp. 1–6, 2014.

[20] ONF, "Archives - Open Networking Foundation," 2019. [Online]. Available: https://www.opennetworking.org/software-defined-standards/archives/. [Accessed: 21-Jun-2019].

[21] G. de trabajo de ingeniería de I. (IETF), "Protocolo de configuración de red (NETCONF)." .

[22] A. Fernanda, A. Rojas, A. Fernanda, and A. Rojas, "Prototipo del protocolo NETCONF para la Gestión Integrada de elementos de red Prototipo del protocolo NETCONF para la Gestión Integrada de elementos de red," 2013.

[23] M. Merelo Hernández, A. J. Estepa, and A. Profesor Titular, "Estudio de la gestión de configuración a través de NETCONF," 2017.

[24] A. D. Las Villas, "Combinación de mecanismos mpls en una arquitectura sdn," in Convención Científica de Ingeniería y Arquitectura, 2018, pp. 1–8.

[25] Francisco José Ibáñez García, "Estudio de las tecnologías SDN y NFV," 2016.

[26] M. Ramirez Giraldo, "Validacion del Nuevo Paradigma de Redes de Datos Definidas por Software de

Tablas de Enrutamiento (MPLS) y Enrutamiento Basado en Políticas," 2017.

[27] J. Duffy, "Cisco reveals OpenFlow SDN killer OpFlex protocol for ACI offered to IETF, OpenDaylight," Networkworld, 2014. .

[28] CISCO, "OpFlex: An Open Policy Protocol White Paper." .

[29] G. de trabajo de ingeniería de I. (IETF) D. Farinacci, "El Localizador / Protocolo de Separación de ID (LISP)," IETF, pp. 11–12, 2013.

[30] A. C.-A. Alberto Rodriguez-Natal, Marc Portoles-Comeras, Vina Ermagan, Darrel Lewis, Dino Farinacci, Fabio Maino, "LISP: a southbound SDN protocol?," ResearchGate, p. 5, 2015.

[31] M. McNickle, "Cinco protocolos SDN que no son OpenFlow," TechTarget, 2014. .

[32] G. Guerrero, "Desarrollo de una plataforma para evaluar calidad de servicios (QOS) en redes definidas por software (SDN)," 2014.

[33] "Serie SDN Tercera parte: NOX, el controlador original de OpenFlow: la nueva pila." [Online]. Available: https://thenewstack.io/sdn-series-part-iii-nox-the-original-openflow-controller/. [Accessed: 11-Jul-2019].

[34] Techtarget network, "NOX," 2013. .

[35] J. L. Leyton Portilla, "Diseño de una red definida por software empleando una solución basada en software, para la infraestructura de cloud de la Facultad de Ingeniería en Ciencias Aplicadas (FICA)," vol. 1, no. 1, pp. 1–13, 2017.

[36] O. Salman, I. H. Elhajj, A. Kayssi, and A. Chehab, "SDN controllers: A comparative study," Proc. 18th Mediterr. Electrotech. Conf. Intell. Effic. Technol. Serv. Citizen, MELECON 2016, no. April, 2016.

[37] "Controlador OpenFlow de Floodlight - Proyect Floodlight." [Online]. Available: http://www.projectfloodlight.org/floodlight/. [Accessed: 11-Jul-2019].

[38] G. Salinas, "Estudio de redes definidas por software e implementación de escenarios virtuales de prueba," p. 98, 2017.

[39] "What is Ryu Controller? — SDxCentral .com." [Online]. Available: https://www.sdxcentral.com/networking/sdn/definitions/what-is-ryu-controller/. [Accessed: 11-Jul-2019].

[40] S. Asadollahi, B. Goswami, and A. Gonsai, "Software Defined Network, Controller Comparison," Ijircce, vol. 5, no. 2, 2017.

[41] ITU, "Update of the SDN standardization activity roadmap," 2017.

[42] R. Á. Pinilla, "Estudio de las redes definidas por software mediante el desarrollo de escenarios virtuales basados en el controlador opendaylight," Universidad Politécnica de Madrid, 2015.

[43] B. Ribes, "OpenDaylight SDN controller platform," no. October, p. 190, 2015.

[44] J. L. G. Rodriguez Farfán, "Integracion de redes IP utilizando SDN," Inst. Tecnológico Buenos Aires, 2017.

[45] C. Salman, Ola;Imad, Elhaj; Ayman, Kayss;Ali, "SDN controllers: A comparative study," ResearchGate, 2016.

[46] J. Pedro and M. Ortiz, "Implementación de un clúster-controlador de sdn basado en un framework de software libre para la infraestructura cloud de la facultad de ingeniería en ciencias aplicadas," 2018.

[47] "Controladores SDN, elementos para su selección y evaluación.," Rev. Telem@tica, vol. 13, no. 3, pp. 10–20, 2014.

[48] David, "VNX," 2017. [Online]. Available: http://web.dit.upm.es/vnxwiki/index.php/Main_Page. [Accessed: 11-Jul-2019].

[49] "EstiNet - Simulador | EstiNet." [Online]. Available: http://www.estinet.com/ns/. [Accessed: 17-Jul-2019].

[50] David Bombal and J. Duponchelle, "Primeros pasos con GNS3." 2019.

[51] Corporativo, "¿Qué es OMNeT ++?," 2019. [Online]. Available: https://omnetpp.org/intro/. [Accessed: 18-Jul-2019].

[52] NS3, "ns-3 Direct Code Execution (DCE) Manual Direct Code Execution project," vol. 1.9, p. 91, 2016.

[53] "Real Academia de Ingeniería," Latizal, 2017. [Online]. Available: http://www.raing.es/es. [Accessed: 19-Jun-2019].

[54] CISCO, "Protocolo de resolución de direcciones," 2018. [Online]. Available: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipaddr_arp/configuration/15-s/arp-15-s-book/Configuring-Address-Resolution-Protocol.html.

[55] Cisco, "ICMP," 13714-43, 2017. .

[56] Cisco, "Protocolo CDP y LLDP," 3404827, 2019. [Online]. Available: https://community.cisco.com/t5/documentos-routing-y-switching/protocolo-cdp-y-lldp/ta-p/3404827.

[57] Cisco, "Descripción TCP/IP," 13769, 2015. [Online]. Available: https://www.cisco.com/c/es_mx/support/docs/ip/routing-information-protocol-rip/13769-5.html.

[58] Efort, "RTP y RTCP 1 RTP y RTCP 2 RTP (Real-time Transport Protocol) 3 Mixer y Translatorhttps://sites.google.com/site/redeslocalesyglobales/6-arquitecturas-de-redes/6-arquitectura-tcp-ip/9-protocolos-tcp-ip/protocolos-de-nivel-de-red/protocolo-arp," pp. 1–9, 2011.

Descubre tu próxima lectura

Si quieres formar parte de nuestra comunidad,
regístrate en **https://www.grupocompas.org/suscribirse**
y recibirás recomendaciones y capacitación

compÁs

Grupo de capacitación e investigación pedagógica

@grupocompas.ec
compasacademico@icloud.com

compAs
Grupo de capacitación e investigación pedagógica

@grupocompas.ec
compasacademico@icloud.com

ISBN: 978-9942-33-199-1

@grupocompas.ec
compasacademico@icloud.com

compAS
Grupo de capacitación e investigación pedagógica