Diseño de una guía teórica y práctica sobre redes definidas por software (SDN) para la Universidad Santiago de Cali

Juan Mateo Cardona Aguirre[1]
Juan.cardona12@usc.edu.co

Universidad Santiago de Cali, Facultad de Ingeniería, Programa de Ingeniería Electrónica (1)

Resumen

Las redes definidas por software (SDN) son un conjunto de técnicas de comunicación que se han desarrollado para la construcción de una arquitectura de red que divide el control y los datos para la obtención de redes programables, flexibles y automatizables que incrementen sus funcionalidades, independizándose de la infraestructura física. Debido a que las SDN aún se encuentran en etapa de investigación y desarrollo, la mejor opción actual para experimentar sus funcionalidades es llevar a cabo una simulación de la red que se desea implementar a través de un emulador y en este trabajo se usará el simulador **Mininet**, siendo este un software que crea una red virtual ajustada a parámetros reales. Los programas de Ingeniería de la Universidad Santiago de Cali no cuentan en sus currículos con temáticas donde se estudien las SDN, por lo tanto, se pretende desarrollar una guía teórico-práctica sobre redes definidas por software que sirva como base para las futuras prácticas que se efectúen. De tal modo, se construyó una guía de instalación y configuración sobre **Mininet** donde se expone el paso a paso para establecer las conexiones iniciales para simular redes, complementado por un documento donde se enseña el marco teórico actual relacionado a las SDN y culmina con la práctica de algunos comandos generales del **Mininet**, la línea de comandos CLI y la administración de switchs OpenFlow.

Palabras Clave: SDN, **Mininet**, OpenFlow, redes, USC.

Abstract

Software-defined networking they are a set of communication techniques that have been developed for the construction of a network architecture that allows you to divide the control and the data for the production of programmable networks flexible and automated to increase its capabilities, independent of the physical infrastructure. Since the SDN are still at the stage of research and development, the best current option to experience its functionalities is to carry out a simulation of the network that you want to implement, through an emulator called **Mininet**, this being a software that creates a virtual network set to actual parameters. Engineering of the Universidad Santiago de Cali programs do not have in their curricula with topics where consideration the SDN, it therefore intends to develop a theoretical and practical guide to software defined networking that serves as a basis for future practices that are carried out. That way an installation and configuration guide accurate about **Mininet** explaining the development for establishing the initial connections to simulate, in addition to a document where the current theoretical framework related to the SDN is taught and ends with the practice of some generals of the **Mininet** commands, and CLI command line administration of switchs OpenFlow.
Keywords: SDN, **Mininet**, OpenFlow, networks, USC.

I. INTRODUCCIÓN

A través de los años, el internet se ha considerado como una herramienta útil que facilita la realización de actividades cotidianas, además brinda una aproximación de comunicación mediante redes sociales, donde se manifiesta un aumento

Nota: Esta guía de instalación fue construida con base a prácticas sobre un sistema operativo Windows, pero no se limita al lector si usa un sistema operativo diferente. Se aclara que el emulador varía su configuración dependiendo de la práctica y la forma de ejecución.

en la demanda por parte de los usuarios; razón por la cual ha sido sujeto de innovaciones y actualizaciones constantes que satisfagan las exigencias presentadas. Es por ello que se buscan diversas alternativas que suplan las necesidades de controles de acceso, abstracción, escalabilidad, flexibilidad de la red, procesamiento de datos, mejora de seguridad, capacidad de adaptación y automatización de procesos en la entidad.

Lo anteriormente mencionado obliga a explorar nuevas posibilidades que tracen un camino en el cual se impulse el desarrollo de nuevas tecnologías de red, en este caso, redes definidas por software (SDN) las cuales permiten resolver problemas de flexibilidad, seguridad y automatización del proceso (Casado, 2007). Este trabajo de grado es realizado para iniciar la exploración de las SDN y simular las primeras prácticas posibles para la familiarización del entorno de red. Además de simular, se busca medir y comparar los tiempos de retraso y fluctuación frente a otros protocolos de red convencionales.

Al ser un área que se encuentra en desarrollo e investigación, se refleja una escasa existencia de referencias en la implementación SDN. En el pregrado de Ingeniería Electrónica de la Universidad Santiago de Cali no existe un contenido programático sobre redes definidas por software, este documento pretende ser un apoyo teórico- práctico que apoye futuras investigaciones sobre las SDN. De la misma manera, se hace una aproximación al funcionamiento de las SDN a través de diferentes laboratorios de práctica.

II. METODOLOGÍA

APROPIACIÒN CONCEPTUAL

Las SDN proponen una arquitectura donde el plano de control (la inteligencia de los dispositivos de red) y el plano de reenvío (responsable del envío de los paquetes) se encuentran desacoplados, separando así las funciones de gestión de red y de conmutación de paquetes. Uno de los componentes más interesantes de la arquitectura SDN son las Interfaces de Programación de Aplicaciones -Application Programming Interface (API)-. La incorporación de estas API hace posible que los administradores puedan desarrollar sus propias aplicaciones SDN con relativa facilidad y que mediante ellas configuren, administren, aseguren y optimicen los recursos de la red de acuerdo a sus necesidades. SDN proporciona dos elementos de mucha importancia: APIs que permite a los usuarios de la nube construir y gestionar sus redes virtuales en los data centers, y el protocolo Openflow, que puede ser utilizado para la provisión de las tablas de enrutamiento y la funcionalidad básica de conmutación que utilizan mediante los switchs que permiten la conexión del tráfico entre máquinas virtuales e interfaces físicas. El camino de datos (datapath) del switch OpenFlow radica en una tabla de flujo y otra acción que se asocia con la entrada de flujo, el conjunto de acciones que soporta un switch OpenFlow es extensible, aunque se establecen mínimos requisitos para la totalidad de switchs. Se Para implementar el protocolo OpenFlow se debe partir de una premisa, debe contarse con un Switch OpenFlow, cuyos básicos requerimientos se proceden a explicar: El switch OpenFlow tiene 3 partes, una Tabla de Flujo, en primer lugar, con acción asociada a cada entrada de flujo, la cual indica al switch la forma como procesarlo; en segundo lugar, canal seguro que permite conectar al switch con el proceso de control remoto, esto recibe el nombre de controlador (controller), con ello permite que los comandos y paquetes se puedan intercambiar entre el switch y controlador por medio del protocolo OpenFlow, que proporciona camino abierto y estandarizado para que se produzca la comunicación en cuestión. Ante ello, por medio del uso del protocolo, el controlador puede actualizar, añadir y borrar las entradas de flujo, de forma reactiva al igual que proactiva.

Los controladores SDN son el motor de las acciones o políticas que se deben tomar en una red SDN. Un controlador ofrece servicios que pueden realizar un plano de control distribuido que se han creado con código abierto ya sea por cualquier usuario incluyendo estudiantes, investigadores, gobierno y entidades privadas afines a las telecomunicaciones. La diferencia entre cada controlador está en el lenguaje de programación y la plataforma en la que éste trabaje, pero la funcionalidad debe ser la misma ya que todos deben transmitir y recibir paquetes OpenFlow para comunicarse con otros dispositivos SDN.

Como resultado de lo antes expuesto existe una amplia gama de herramientas, tanto comerciales como de código abierto, que están a disposición de los desarrolladores de aplicaciones SDN, así como numerosas guías y tutoriales que los fabricantes ofrecen para la utilización de dichas herramientas.

Para que las redes definidas por software puedan ser implementadas, se necesita de entornos computacionales que fundamenten las condiciones mínimas necesarias para automatizar el control de los datos. Para esto, el usar un emulador como **Mininet** permite crear el ambiente ya descrito y poder observar los comportamientos y características principales de las SDN. En el mercado actual existen algunos softwares y controladores que simulen entornos para SDN; A continuación, se describen los desarrolladores con los cuales se puede emular una red programable.

SIMULADORES PARA SDN		
NOMBRE	LENGUAJE	DESCRIPCIÓN
Mininet	C/Python	Utiliza varias capas virtuales, licencia del código de apache 2.0. Diseñado para permitir la automatización de la red, apoya las interfaces de gestión y protocolos como Netflow, Sflow, SPAN, entre otros. Este código es abierto y gratuito.
Pica8	C	Plataforma de software abierto para el interruptor de hardware de conmutación chips que incluye desarrollo de L3 L2 y soporte para OpenFlow.
Indigo	C	Implementación de OpenFlow con código abierto el cual funciona con características similares a **Mininet**.
Pantou	C	Plataforma de software abierto para cualquier tipo de hardware de conmutación, el desarrollo de OpenFlow está implantado en cuanto a una aplicación llamada OpenWrt, la cual es una distribución de Linux.
OpenFaucet	Python	Aplicación Python pura con OpenFlow 1, se puede utilizar solo dos interruptores.
OpenFlowJ	Java	Desarrollado por Java, solo contiene una versión del OpenFlow hasta la fecha.
Oflib-node	Javascript	Es una biblioteca de protocolo OpenFlow para el nodo. Convierte los mensajes del protocolo OpenFlow en objetos de Javascript.
Nettle	Haskell	OpenFlow escrito en Haskell.

Para el desarrollo de los laboratorios el emulador a utilizar será **Mininet** puesto que es una plataforma de pruebas de red de código abierto, rápido y amigable con entornos de diferentes topologías. Hasta ahora, es la herramienta de apoyo a la investigación de las SDN OpenFlow más conocida, como se denotó en la Conferencia Mundial de Desarrollo de las Telecomunicaciones del año 2015 (Telecomunicaciones, 2015), conferencia que hace parte de la Organización Internacional de Telecomunicaciones (OIT).

2.2 Construcción de un documento teórico para la enseñanza de los conceptos básicos de las redes definidas por software (SDN)

Según (Stallings, 2015) los documentos teóricos sobre SDN deben asociar tópicos importantes como: antecedentes sobre anteriores protocolos, comparativa con sistemas tradicionales, separación del plano de control y de datos, virtualización de redes, fundamentos de Redes Definidas por Software, Fundamentos OpenFlow y manejo de emuladores compatibles. En la actualidad, son pocos los cursos especializados en las redes definidas por software, algunos de estos ejemplos pueden situarse en: Curso en línea: Coursera, Software Defined Networking apoyada por University Princeton, compuesta por 8 módulos (Nadeau & Gray, 2016). En Colombia la Universidad ICESI de Santiago de Cali ofrece una asignatura llamada Redes Definidas en Software en Ingeniería Telemática (ICESI, 2018) y en ella aborda 3 unidades que van desde los antecedentes y conceptos básicos hasta implementación de SDN en casos prácticos. Esta asignatura podrá servir como punto de partida para la construcción de contenidos temáticos alrededor de las redes definidas por software en la USC.

Teniendo en cuenta esta revisión de cursos ofrecidos, se propone la siguiente estructura para el anexo-guía que servirá como base teórica para la enseñanza de las redes definidas por software:

Nota: Esta guía de instalación fue construida con base a prácticas sobre un sistema operativo Windows, pero no se limita al lector si usa un sistema operativo diferente. Se aclara que el emulador varía su configuración dependiendo de la práctica y la forma de ejecución.

Importancia de las SDN, Antecedentes de las SDN, Marco histórico, Active Networking, Virtualización de las redes, Concepto de abstracción, Planos (Datos y Control), Separación del plano de datos y plano de control, Software-defined networking, Open networking foundation, Ventajas de SDN (Rendimiento vs flexibilidad, escalabilidad, seguridad, interoperabilidad, calidad del servicio), Estructura del SDN (aplicaciones, Controller, Northbound API, East-West API, Southbound API). Operación del SDN, Seguridad, Protocolo OpenFlow, Switch OpenFlow, Puertos OpenFlow, Canal OpenFlow, Mensajes OpenFlow, Comparativa entre versiones de OpenFlow.

Como complemento a este artículo, se consideró crear un documento anexo que servirá para definir la temática a afrontar, de tal manera que se ayuda en la enseñanza de las redes programables por software. Esta organización estructural servirá para abordar la temática de SDN donde se abarca gran cantidad de conocimiento adquirido durante los últimos años por las diferentes personas que han intervenido de manera activa en las SDN.

2.3 Prácticas en un entorno de red virtual para emular el ambiente computacional de las redes definidas por software

Hoy en día es necesario modelar hosts, conmutadores, enlaces y controladores SDN/OpenFlow. Mininet permite crear topologías de gran tamaño de escala hasta miles de nodos y realizar pruebas en ellos muy fácilmente. Tiene herramientas de línea de comandos y API muy simples. Mininet permite al usuario crear, personalizar, compartir y probar redes SDN fácilmente. En la Figura 1 se muestra cómo es la red emulada por Mininet.

La herramienta de emulación **Mininet** tiene como ventaja la capacidad de integrarse con dispositivos reales como switches, OpenFlow físicos y todo tipo de controllers, esto permite el diseño de prácticas que acerquen al estudiante a un ambiente de trabajo de dispositivos de redes virtuales y físicas de códigos abiertos o comerciales (Velasquez Vargas, 2014). Las prácticas de la guía SDN están enfocadas en ilustrar la teoría tratada y como opción de desarrollo de habilidades básicas en las redes programables. Durante el desarrollo del presente trabajo se realizaron tres prácticas que consisten: en el manejo básico del emulador **Mininet**, analizar el protocolo OpenFlow y controlar manualmente los switchs OpenFlow.

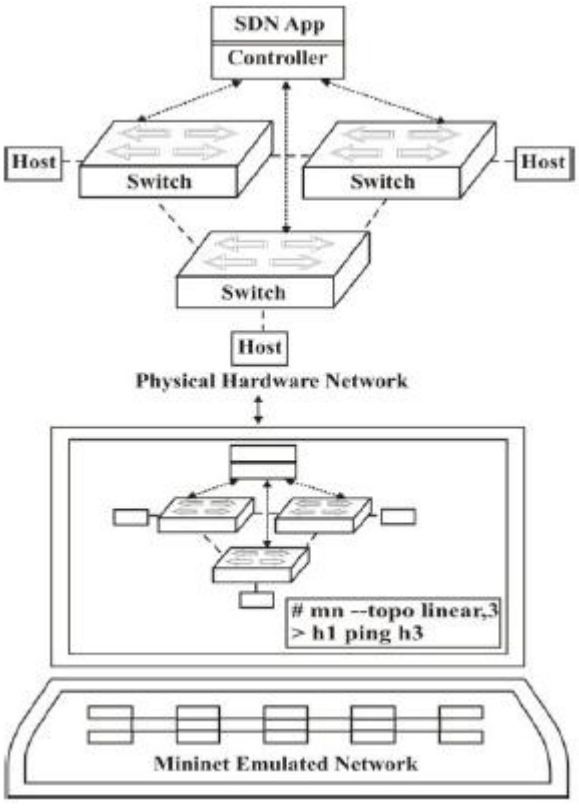
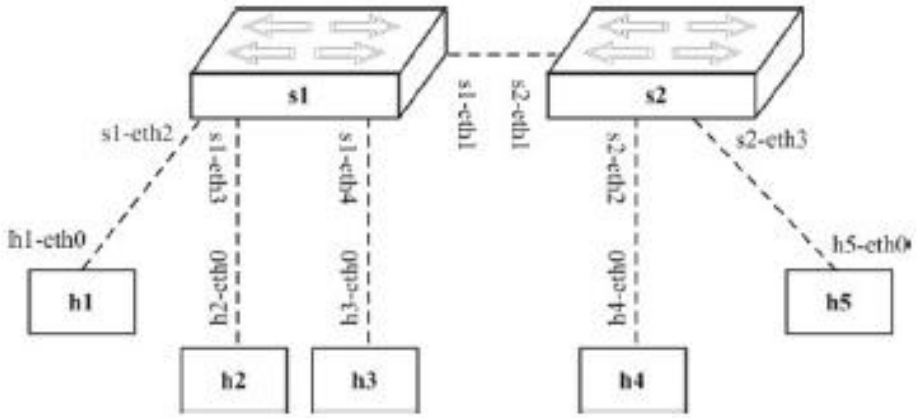


Figura 1: Red emulada a través de Mininet. Fuente: (Kaur, Singh, & Ghumman, 2018)

TOPOLOGÌA DE RED PARA SDN EN MININET

Mininet contiene un número de topologías predeterminadas como mínimal, single, inverse, lineal y tree (Kaur, Singh, & Ghumman, 2018). Comprender el método de nomenclatura para interfaces, hosts y conmutadores es esencial para prosperar con Mininet. Los switchs se nombran de s1 a sN. Los anfitriones se denominan h1 a hN. Las interfaces de host se denominan con el nombre del host seguido por el nombre Ethernet a partir de 0. La primera interfaz del host 'h1' se llama 'h1-eth0' y la tercera interfaz del host 'h2' se llama 'h2-eth2'. El primer puerto del switch 's1' se denomina 's1-eth1'. En los interruptores, la numeración comienza con 1. En las diferentes prácticas sobre SDN la topología de red estará determinada por una cantidad variable de switchs y host, además de su respectivo controlador OpenFlow. En la figura 2 se muestra una topología de red personalizada creada a través del emulador.

Figura 2: Topología de red personalizada en Mininet. Fuente: Lopez & Ramirez, 2018)



A continuación, se detalla el código en Phyton usado para crear en Mininet una red personalizada. En este caso, esta topología está acompañada de 2 conmutadores o switchs y 5 hosts o anfitriones.

```
"from mininet.cli import CLI
from mininet.util import dumpNodeConnections
from mininet.node import CPULimitedHost
from mininet.link import TCLink
class SingleTopologyPerformance (Topo)

        def_init_ (self, k=3);
                Topo. _init_(self)
                Switch=self.addSwitch('s1')
                Linkoptions=dict(bw=10, delay='10ms', max_queue_size=1000, use_htb=true)
                For h in range (k);
                        Host=self.addHost('h%s'%(h+1), cpu=4/k)
                        Self.addLink (host, switch, **linkoptions)
Def performanceTest()
        Topo=SingleTopologyPerformance(k=5)
        Net=Mininet(topo=topo, host =CPULimitedHost, link=TCLink)
        Net.start()
        Print "Mostrando información de conexión de los anfitriones"
        DumpNodeConnections(net.hosts)
        Print "Verificando el ancho de banda entre host h1 y h3"
H1,h3 = net.get('h1','h3')
Net.iperf(('h1','h3'))
Net.stop()
If_name_ =='_main_';
        setLogLevel ('info')
        performanceTest()
```

Nota: Esta guía de instalación fue construida con base a prácticas sobre un sistema operativo Windows, pero no se limita al lector si usa un sistema operativo diferente. Se aclara que el emulador varía su configuración dependiendo de la práctica y la forma de ejecución.

2.3.1 Principios generales del emulador Mininet

Esta práctica tiene como objetivo explorar el ambiente de trabajo del emulador **Mininet** en aspectos tales como, el uso de comandos básicos, uso de herramientas de terceros, parametrización de las emulaciones, entre otros. Para la ejecución de este laboratorio sólo se necesita un computador con el software previamente instalado, como se detalla anteriormente en la guía de instalación del emulador **Mininet** a través de una virtualización de la red. Por medio de este entrenamiento se busca implementar instrucciones esenciales como **sudo mn** para la creación de una red en la que su topología esté definida según las especificaciones requeridas <linear, minimal, single, reversed o tree> (Santos de Oliveira, Schweitz, Akira Shinoda, & Rodrigues Prete, 2014), además de una cantidad considerable de comandos necesarios para ampliar los conocimientos elementales sobre la aplicación. La estructura del comando sudo mn en **Mininet** se define de la siguiente manera:

Sudo mn – [OPCIÓN] = [PARÁMETRO], [ARGUMENTOS] – [OPCIÓN_n] = [PARÁMETRO_n], [ARGUMENTOS]

Para empezar a interactuar con el entorno de **Mininet** se pretende usar la línea de comandos CLI y ver algunas características de la red. Se inicia con el comando **Mininet**> dump para mostrar los equipos que forman parte de la red e información sobre los mismos. Continua **Mininet**> xterm <<nombre del equipo>> para abrir un terminal independiente del equipo deseado (este comando exigirá ingresar el nombre del dispositivo). Para ver la conectividad de la red y comprobar que funciona adecuadamente se usará el comando **Mininet**> pingall, esta línea despliega la cantidad de host en la red y cómo se comportan funcionalmente entre ellos. En el CLI se van a ejecutar las líneas de código más usadas en **Mininet** a la hora de simular una red: exit, help, net, intfs, nodes, ports, time, switch, links, link, noecho, sh, source, iperf, px, py, xterm y dpctl.

Para el control manual de los switchs OpenFlow se usará la herramienta dpctl con sus opciones y argumentos (Bianco, Birke, & Palacin, 2010). Los comandos de administración de Switchs OpenFlow brindan información específica sobre el switch, imprime en pantalla las características de parámetros del switch, muestra estadísticas y modifica el comportamiento del puerto. En el desarrollo de esta práctica se utilizan los comandos show, status, show-protostat, dum-desc, dump-ports, dump-flows, monitor, probe, ping, benchmark, add-flow, mod-flows y del-flows. La prueba de cada uno de estos comandos servirá para familiarizarse con el protocolo OpenFlow. La estructura de dpctl es la siguiente:

Dpctl [OPCIONES] COMANDO [SWITCH [ARGUMENTOS]]

Para el desarrollo de las pruebas de envío de paquetes de datos con diferentes bytes, es necesario configurar el tipo de topología a implementar, así mismo como sus preferencias y el direccionamiento del host a través de IPV6:
Se reinicia el OpenVswitch con el comando: ***/etc/init.d/openvswitch-switch restart.*** Se añade un puente y después una interfaz física realizando un puente virtual de la siguiente manera:
ovs-vsctl add-br br-int
ovs-vsctl add-port br-int eth0
ifconfig eth0 0

A la interfaz eth0 se le configura un puente de la siguiente manera: ***ifconfig br-int 192.168.1.208 netmask 255.255.255.0***, de esta manera cambiará la ruta que viene por defecto: ***route add default gw 192.168.1.1 br-int and route del default gw 192.168.1.1 eth0***. Ahora, para los controladores SDN se instalará el POX de la siguiente forma:
Sudo apt-get install git
git clone http://github.com/noxrepo/pox 88
Cd pox
. /pox.py forwarding.l2_learning
NOX>
Se puede observar que por defecto se une al puerto 6633 y los demás.
Por último, se tendrá que adjuntar el OpenVswitch al controlador por medio del comando: ***ovs-vsctl set-controller br-int tcp: 192.168.1.208:6633.***
Realizado estos pasos ya se tendrá la maquina funcionando con OpenVswitch y con el controlador POX.

Para continuar el desarrollo de las pruebas se prosigue a configurar las propiedades de cada host, para esto se procede a configurar la red de la siguiente manera: Con click derecho mantenido en el host 1, se entra a propiedades y se configura las IP en IPv4 para cada uno de los hosts virtuales creados. Para el desarrollo de IPv6 se ejecuta por comando ya que la

versión del **Mininet** no está implementado en modo grafico por lo tanto lo se realizará de la siguiente manera con el comando ***ifconfig h1-eth0 inet6 add fc00:a:1/64*** tanto para host 1 como para host2.
Por otro lado, se configura el controlador con el puerto por defecto 6633, y el tipo de controlador en este caso OpenFlow Reference y la IP de loopback 127.0.0.1. Por último, se editan las configuraciones de preferencia del host como se muestra en la Figura 3.

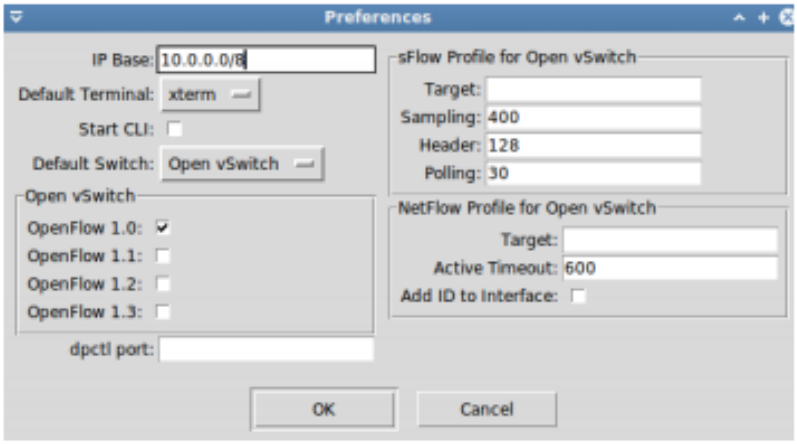


Figura 3. Configuración de las preferencias del host. Fuente: (Sombredero & Siva, 2015)

MEDICION DE JITTER Y DELAY EN ENVIO DE PAQUETES

Para finalizar el laboratorio se procede a realizar las pruebas de 10, 100, 6000 y 30000 Bytes con 200 paquetes cada una. La prueba consiste en el envío de paquetes anteriormente mencionados para proceder a realizar la medición de Delay y Jitter y compararlos con otra tecnología de redes como las convencionales, esta medición se hará bajo los estándares IpV4 e IpV6. Por otro lado, el desarrollo de la medición (jitter y delay) estará enfocado a un desempeño pasivo, estos solo capturan el tráfico que pasa a través de una interfaz de red (Velazquéz E, 2009), por último, según el análisis en el marco teórico el jitter y delay son variables válidas para medir el desempeño en las redes y pueden ser usadas para realizar la comparación.

Nota: Esta guía de instalación fue construida con base a prácticas sobre un sistema operativo Windows, pero no se limita al lector si usa un sistema operativo diferente. Se aclara que el emulador varía su configuración dependiendo de la práctica y la forma de ejecución.

III. RESULTADOS Y DISCUSIÓN

3.1. Mininet

Una vez descargada la máquina virtual (Virtual Box) se importó el archivo de **Mininet**, siendo necesario configurar el adaptador a un sólo anfitrión. Es primordial verificar que exista una interfaz entre **Mininet** y el entorno virtual, de no ser así, crearla. Este adaptador demanda estar habilitado el servicio DHCP. Al momento de iniciar la máquina, se ingresan los datos del usuario y la contraseña, procedido a esto se habilita la conexión SSH por medio de PuTTY (Llaudet Planas, 2003). Para configurar independientemente cada host virtualizado en **Mininet**, es preciso servirse de una herramienta llamada Xming, el software que reconoce el acceso al programa Whireshark para realizar capturas de paquetes de la emulación. Para usar este servicio es necesario correrlo en el sistema, más tarde se conecta del SSH a la máquina virtual por medio de Putty, al que anteriormente se le activa la casilla "Enable X11 forwarding" ubicado en "Connection" /"SSH" /"X11" de la configuración de PuTTY. Cuando se concluye la configuración, se inicia la ventana de comandos **Mininet** CLI con el comando "sudo mn" en el terminal puTTY y este a su vez crea la red, con sus respectivos controladores y host como se puede apreciar en la Figura 2. Una vez que la línea de comandos esté lista para iniciar la red, se debe verificar que las aplicaciones Whireshark y Xming se estén ejecutando en la máquina virtual ya que sirven como software de soporte a la nueva red en esta simulación. En la Figura 4 se demuestra la conexión exitosa entre el emulador y sus complementos.

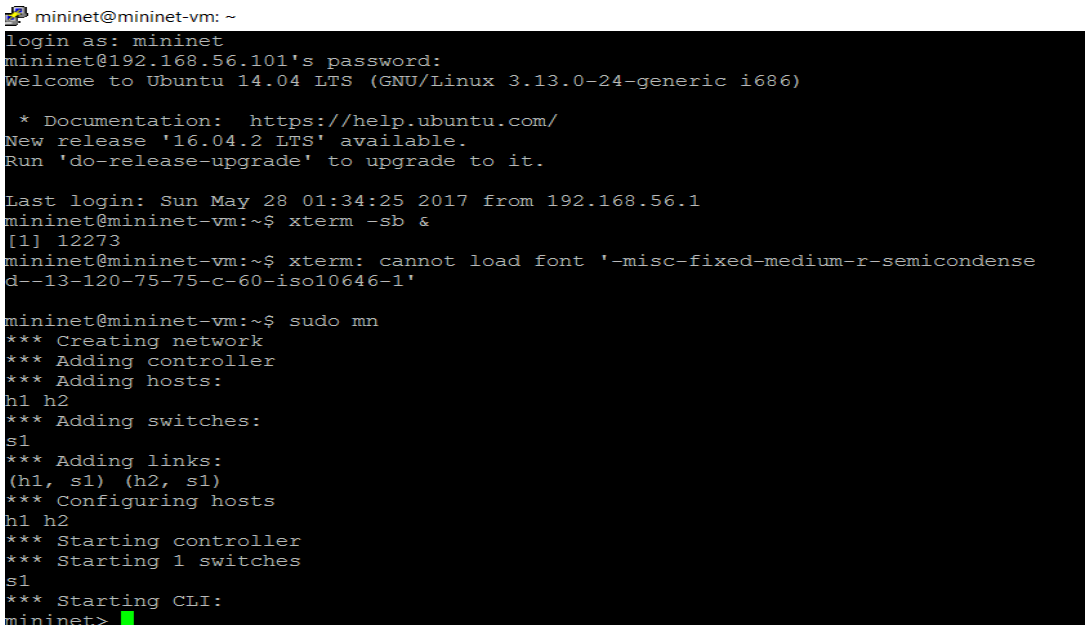


Figura 4. Ventana de comandos CLI en **Mininet**. Fuente propia

3.2 Prácticas en un entorno de red virtual usando el emulador Mininet

3.2.1 Principios generales de Mininet

Al iniciar **Mininet** se invocan constructores que parametrizan por defecto la emulación, adicionalmente pueden ser personalizados según las necesidades presentadas y la finalidad del ejercicio. Para iniciar la interacción con este software, se recomienda abordar los comandos sudo mn puesto que son la base de cualquier simulación de red. En la Figura 5 se muestran las utilidades del comando sudo mn. En el desarrollo del laboratorio sobre principios generales del emulador es recomendable familiarizarse con cada opción de topología puesto que las opciones que ofrece el programa son múltiples y su uso depende de la finalidad.

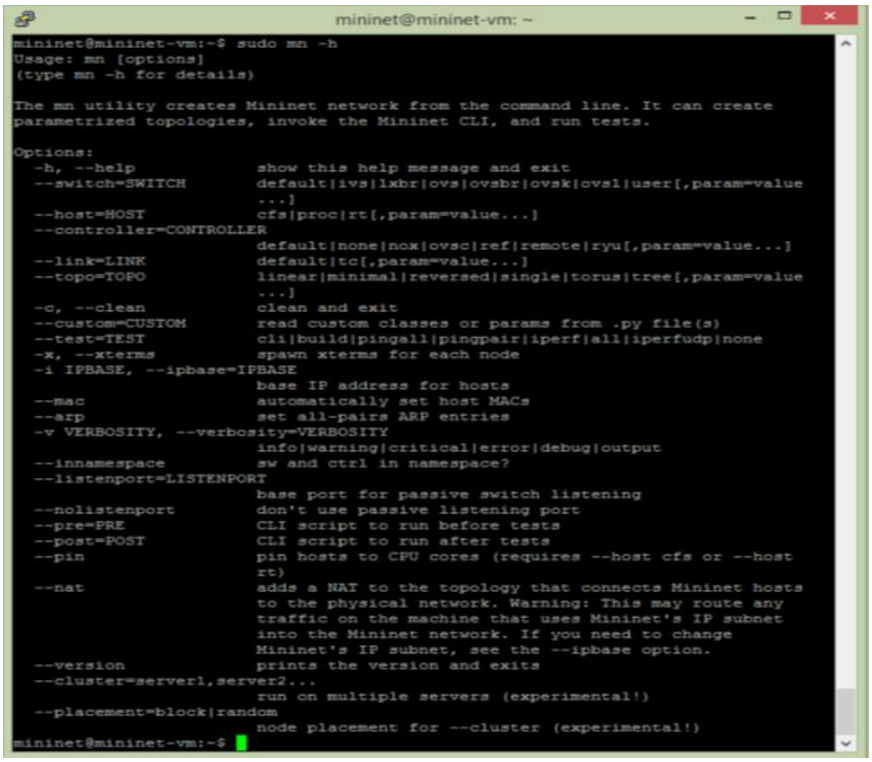


Figura 5. Salida del comando sudo mn help. Fuente propia.

En **Mininet** algunas topologías disponibles para emulación son: minimal, single, linear, tree y personalizadas. En cualquiera de éstas existe un controlador, el número de hosts varía al igual que los switchs y enlaces entre éstos. Minimal, por ejemplo, está compuesta por dos hosts conectados a un switch; single también es una topología con un único switch pero la cantidad de host puede indicarse por parámetro. En la Figura 6 se observa una red con topología tree o árbol que puede ser emulada por el software, para ejecutarla se utiliza la instrucción sudo mn --topo tree, depth=n donde n es la profundidad en niveles de la red. En el caso de la topología tratada, la profundidad escogida es 3.

En este caso **Mininet** crea 8 hosts (2 conectados a cada uno de los switches del nivel más inferior) y en total 7 switches. La Figura 6 presenta la ejecución de esta topología y el comando links con el que pueden comprobarse las conexiones entre todos los equipos y dispositivos emulados. Las topologías personalizadas, por su parte, permiten el diseño de redes con las conexiones y la cantidad de dispositivos específicos deseados. Estas topologías deben implementarse en scripts de Python utilizando las librerías y la API (Hata, 2013) que para este fin tiene **Mininet**. La ejecución de los comandos en el emulador para recrear una red tipo árbol sería de la siguiente manera:

Mininet> net
H1 h1-eth0: S3-eth1
H2 h2-eth0: s3-eth2
H3 h3-eth0: s4-eth1
H4 h4-eth0: s4-eth2
H5 h5-eth0: s6-eth1
H6 h6-eth0: s6-eth2
H7 h7-eth0: s7-eth1
H8 h8-eth0: s7-eth2
S1 lo: s1-eth1: s2-eth3 s1-eth2: s5-eth3
S2 lo: s2-eth1: s3-eth3 s2-eth2: s4-eth3 s2-eth3: s1-eth1
S3 lo: s3-eth1: h1-eth0 s3-eth2: h2-eth0 s3-eth3: s2-eth1
S4 lo: s4-eth1: h3-eth0 s4-eth2: h4-eth0 s4-eth3: s2-eth2
S5 lo: s5-eth1: s6-eth3 s5-eth2: s7-eth3 s5-eth3: s1-eth2
S6 lo: s6-eth1: h5-eth0 s6-eth2: h6-eth0 s6-eth3: s5-eth1
S7 lo: s7-eth1: h7-eth0 s7-eth2: h8-eth0 s7-eth3: s5-eth2

Nota: Esta guía de instalación fue construida con base a prácticas sobre un sistema operativo Windows, pero no se limita al lector si usa un sistema operativo diferente. Se aclara que el emulador varía su configuración dependiendo de la práctica y la forma de ejecución.

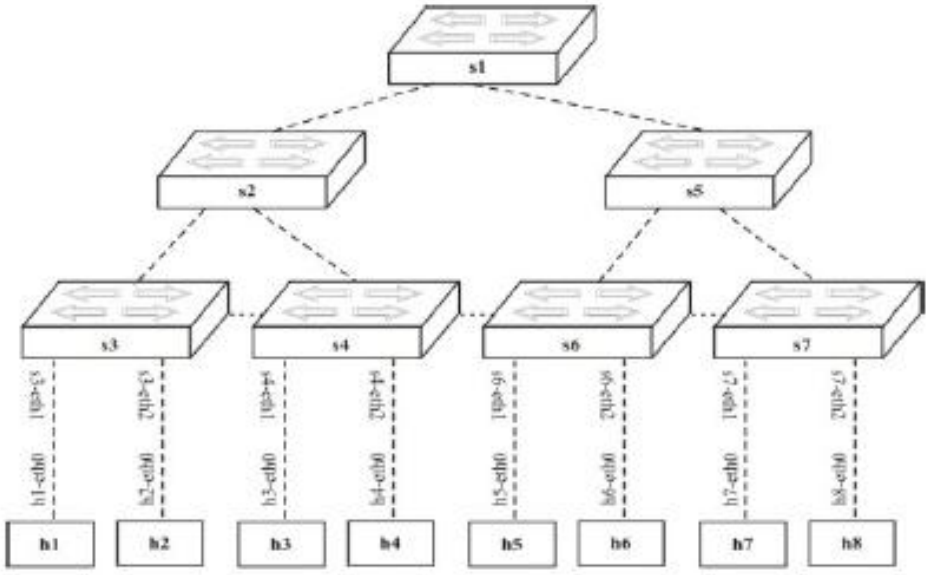


Figura 6. Topología tree o árbol para emulación en **Mininet**. Fuente: (Kaur, Singh, & Ghumman, 2018)

```
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7
*** Adding links:
(s1, s2) (s1, s5) (s2, s3) (s2, s4) (s3, h1) (s3, h2) (s4, h3) (s4, h4) (s5, s6)
 (s5, s7) (s6, h5) (s6, h6) (s7, h7) (s7, h8)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 7 switches
s1 s2 s3 s4 s5 s6 s7
*** Starting CLI:
mininet> links
s1-eth1<->s2-eth3 (OK OK)
s1-eth2<->s5-eth3 (OK OK)
s2-eth1<->s3-eth3 (OK OK)
s2-eth2<->s4-eth3 (OK OK)
s3-eth1<->h1-eth0 (OK OK)
s3-eth2<->h2-eth0 (OK OK)
s4-eth1<->h3-eth0 (OK OK)
s4-eth2<->h4-eth0 (OK OK)
s5-eth1<->s6-eth3 (OK OK)
s5-eth2<->s7-eth3 (OK OK)
s6-eth1<->h5-eth0 (OK OK)
s6-eth2<->h6-eth0 (OK OK)
s7-eth1<->h7-eth0 (OK OK)
s7-eth2<->h8-eth0 (OK OK)
mininet>
```

Figura 7. Ejecución del comando links en una topología tree. Fuente propia.

Usando uno de estos parámetros es posible recrear un tipo de red diferente dependiendo del uso que se le quiera dar y la configuración deseada por el programador (Feamster, Rexford, & Zegura, 2013). Además de la configuración de topología, también existen otros comandos que reconocen distintas funciones para simular la red. Para evaluar cada situación de estos comandos se probaron las instrucciones "switch" para invocar un tipo de switch específico, "controller" para invocar un tipo de controller en específico, "links" (como se muestra en la Figura 7) para verificar conexión entre los dispositivos, "clean" para limpiar los registros de emulación. El comando "custom" sirve para leer los archivos de configuración escritos en Python con extensión .py para crear redes personalizadas. Además de las ya mencionados, **Mininet** permite el uso de otras instrucciones para ejecutar pruebas, iniciar la simulación y abrir terminales

independientes en cada dispositivo emulado. Se recomienda interactuar con cada uno de estos comandos para visualizar el alcance del emulador en la creación de la red.

3.2.2 Línea de comando de Mininet

Estas instrucciones están disponibles después de llevar a cabo el comando "sudo mn". Para más información se puede gestionar en la consola **Mininet** help + [COMANDO]. **Mininet** está habilitado para tramitar comandos shell sobre los dispositivos emulados (controllers, switches, host), para esto se debe digitar en primer lugar el nombre del dispositivo, seguido del comando y sus parámetros (Velasquez Vargas, 2014). A continuación, se describen algunos comandos CLI de **Mininet** usados para interactuar con la red:

- **Mininet**> EOF: el comando "EOF" finaliza la emulación de **Mininet**.
- **Mininet**> exit: el comando "exit" termina la emulación y cierra el programa.
- **Mininet**> quit: el comando "quit"" cesa la emulación actual.
- **Mininet**> help: el comando "help" refleja en pantalla documentación e información del uso de comandos.
- **Mininet**> dump: el comando "dump" expone en pantalla información detallada de la red, datos como tipo de dispositivo, nombre, puerto usado, dirección IP e ID de proceso.
- **Mininet**> net: el comando "net" muestra en pantalla los enlaces y los respectivos puertos empleados por los instrumentos emulados.
- **Mininet**> intfs: el comando "intfs" lista las interfaces usadas por los artefactos emulados.
- **Mininet**> dpctl [COMANDO] [ARGUMENTOS]: dpctl (o ovs-ofctl) es una herramienta de administración y monitoreo que obra en todos los switches OpenFlow emulados, cuando se trae desde la consola **Mininet**; limita sus funciones, por lo tanto, guía usa dptcl de forma externa al emulador **Mininet**.

Con el objetivo de simplificar cada una de las opciones y los argumentos, se refiere dentro de la tabla 1 el listado de comandos que pueden servirse en la práctica de implementación del emulador **Mininet**. Cada una de estas opciones tiene una serie de parámetros que reconocen cualquier tipo de configuración deseada a emular como se muestra en la siguiente tabla.

Tabla 1. Lista de comandos. Fuente propia. Fuente: El Autor.

COMANDO	ARGUMENTOS	DESCRIPCIÓN
EOF		Finaliza la emulación.
exit		Finaliza la emulación.
quit		Finaliza la emulación.
help		Muestra información.
dump		Información detallada de la red.
net		Información de enlaces.
intfs		Información de interfaces.
nodes		Nodos usados.
ports		Puertos usados.
time	[COMANDO]	Tiempo de ejecución.
switch	[SW] [start|stop]	Inicia o finaliza el switch.
links		Reporte de enlaces operativos.
links	[NODO1] [NODO2]	Des/habilita enlaces.
noecho	[HOST] [CMD args]	Ejecuta comandos shell en hosts.
sh	[CMD args]	Ejecuta comandos shell en anfitrión.
source	<FILE>	Lee comandos **Mininet** desde fichero.
pingall		Prueba conexión de toda la red.
pingallfull		Prueba conexión y detalles.

Nota: Esta guía de instalación fue construida con base a prácticas sobre un sistema operativo Windows, pero no se limita al lector si usa un sistema operativo diferente. Se aclara que el emulador varía su configuración dependiendo de la práctica y la forma de ejecución.

pingpair		Prueba conexión entre h1 y h2.
pingpairfull		Prueba entre h1 y h2 detallada.
iperf	[HOST1] [HOST2]	Rendimiento BW TCP.
iperudp	[BW] [H1] [H2]	Rendimiento BW UDP.
px	[PYTHON]	Ejecución declaraciones Python.
py	[OBJETO.FUNCION()]	Ejecución expresiones Python.
xterm	[HOSTn]	Abre consolas independientes.
x	[HOST] [CMD args]	Creación de túnel X11.
gterm	[HOSTn]	Abre consola GUI independiente.
dpctl	[COMANDO] [args]	Ejecuta funciones dpctl.

3.2.3 Control manual de switchs OpenFlow

En el emulador **Mininet** se realiza simulación con switchs OpenFlow a través de un comando dpctl. La función 'dpctl' es una herramienta de administración y monitoreo que permite crear, modificar y eliminar entradas de flujo en switchs OpenFlow sin la necesidad de requerir el uso de un controller; esta utilidad sirve para establecer reglas de flujos manuales y puntuales.

La instrucción "dpctl" tiene una estructura definida por una opción, un comando, el switch y un argumento. Cada uno de estos parámetros es usado para ejecutar cualquier configuración que se desee diseñar para la práctica con switchs como se denota a continuación:

[OPCIONES]: Los argumentos pasados en el campo opciones no son obligatorios y son usados para propósitos de información y cambio de comportamiento de los comandos por defecto.

[SWITHC]: El campo switch es obligatorio ya que especifica el método de conexión que se usará con un dispositivo OpenFlow.

[Args…]: Recibe diferentes valores según el comando ingresado.

COMANDO, de uso obligatorio, este parámetro recibe un comando que permite ejecutar funcionalidades como exponer información, manipular tabla de flujo e imprimir en pantalla estadísticas de los parámetros.

El comando "status [SWITCH]" imprime en pantalla estadísticas de parámetros del switch especificado (Bianco, Birke, & Palacin, 2010). Cómo se visualiza en la Figura 8, **Mininet** arroja datos sobre cada uno de los parámetros del Switch OpenFlow; ejemplo, se observan los estados, el número de conexiones y la configuración de cada parámetro.

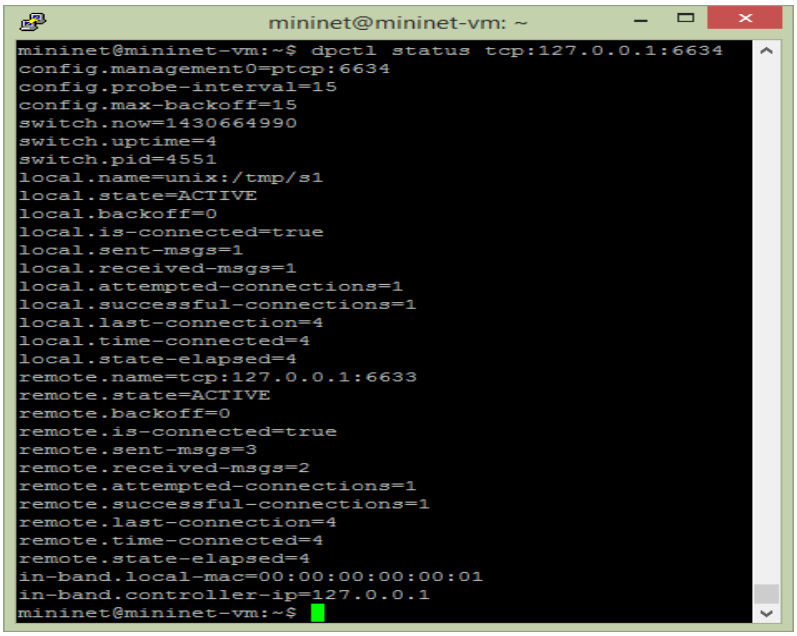


Figura 8. Parámetros del switch especificado vistos con el comando "status [SWITCH]". Fuente Propia

Este tipo de comandos sirven como apoyo en el momento de administrar el switch. Para complementar la información de la variedad de comandos para ejecutar en la administración de los switchs, en la Tabla 2 se muestra cada comando con su respectiva opción y argumento.

Tabla 2. Lista de comandos en la administración de los switchs OpenFlow. Fuente: El Autor

DPCTL	OPCIONES		COMANDO	DISPOSITIVO	ARGUMENTOS	FUNCIONALIDAD
dpctl	timeout	=[SEG]	show status	tcp: IP:PUERTO		Imprime en pantalla estadísticas de parámetros del switch.
	verbose		show-protostat			Estadísticas del protocolo OpenFlow en el switch.
	log-file	<FILE>	dump-desc			Descripción del switch.
	help		dump-tables			Estadísticas de las tablas del switch.
	version		mod-port		[up|down|flood|noflood]	Modifica el comportamiento del puerto en el switch.
			dum-ports		[PUERTO]	Estadísticas de todos los puertos del switch.
			dump-flows		[FLUJO]	Entradas de flujo.
			dump-aggregate		[FLUJO]	Estadísticas adicionales para un flujo de paquetes.
			Monitor			Todos los mensajes OpenFlow recibidos en el switch.
			Probe			Envía paquetes de prueba.
			Ping		[N]	Envía 10 paquetes de [n]-bytes probando conectividad y tiempos
			Benchmark		[N] [CONTADOR]	Envía una cantidad de paquetes [n] + 8 bytes de cabecera.
			add-flow		[FLUJO]	Agrega entradas a las tablas de flujo del switch.
			add-flows		<FILE>	Agrega entradas a las tablas de flujo del switch a partir de un fichero.
			mod-flows		[FLUJO]	Modifica las acciones de una entrada de flujo.
			del-flows		[FLUJO]	Elimina la entrada de flujo en el switch.

RESULTADOS DE PRUEBAS SIMULADAS PARA MEDICION DE JITTER Y DELAY EN SDN IPV4 E IPV6

Las siguientes tablas y gráficas muestran el comportamiento de las pruebas realizadas para la medición de dos parámetros diferentes en las SDN a través del emulador **Mininet**. Esta evaluación se hizo a través de los protocolos Ipv4 e IpV6 para cuatro tamaños de paquetes diferentes. Estos resultados fueron obtenidos por medio de los comandos ***show-protostad*** – opción ***verbase.*** En la Tabla 3 se muestran los datos encontrados en la simulación hecha a través del emulador.

Tabla 3. Prueba de simulación para SDN en IPv4

PRUEBA SIMULADA IPv4					
RED	MEDICIÓN	TAMAÑO (Bytes)			
		10	100	6000	30000
SDN	DELAY (ms)	0,55	0,76	2,04	2,64
	JITTER (ms)	0,44	0,6	1,78	1,74

Nota: Esta guía de instalación fue construida con base a prácticas sobre un sistema operativo Windows, pero no se limita al lector si usa un sistema operativo diferente. Se aclara que el emulador varía su configuración dependiendo de la práctica y la forma de ejecución.

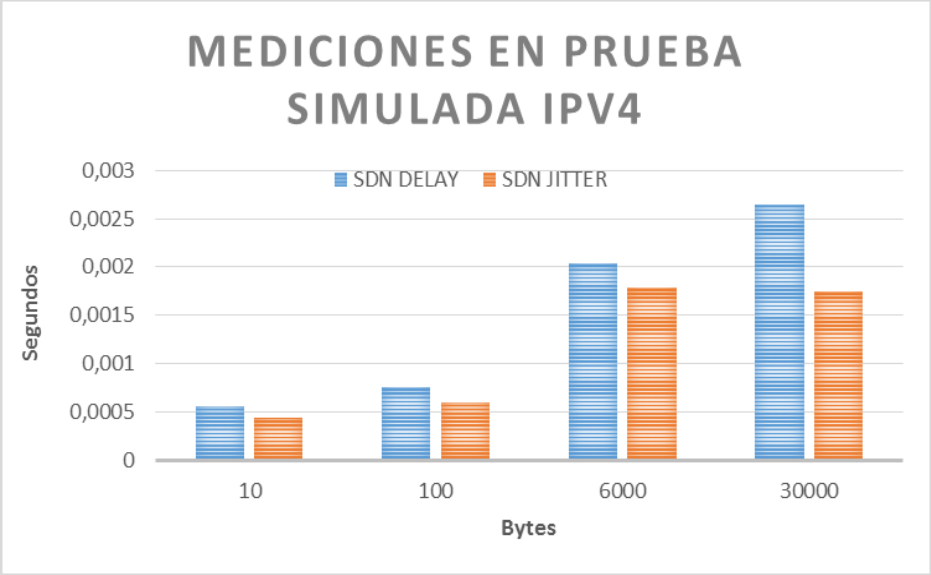


Figura 9: Gráfica comparativa medición de delay y jitter en SDN IpV4. Fuente propia

Las mediciones que se realizaron para SDN IPv4 detallan valores bajos para los retrasos y las fluctuaciones que se querían medir en comparación a otro tipo de tecnologías de redes convencionales como OSPF y MPLS. Según (Sombredero & Silva, 2015) el protocolo de red Open Shortest Path First (OSPF), Primer Camino Más Corto, tiene Delay para 10, 100, 6000 y 30000 bytes los siguientes valores: 12.38, 10.68, 11.53 y 13.23 respectivamente. Para la prueba de Jitter los resultados encontrados por (Sombredero & Silva, 2015) fueron de 12.94, 11.32, 10.43 y 7.75 respectivamente. Comparando estos resultados encontrados frente a los de SDN, se observan valores de errores muy bajos para las redes definidas por software frente a otros protocolos de red.

COMPORTAMIENTO DEL DELAY Y JITTER PARA SDN EN IPV6

Tabla 4. Prueba de simulación para SDN en IPv6

PRUEBA SIMULADA IPv6					
RED	MEDICIÓN	TAMAÑO (Bytes)			
		10	100	6000	30000
SDN	DELAY (ms)	2,55	1,9	3,58	7,18
	JITTER (ms)	2,43	0,85	3,4	5,35

Los resultados presentados en la Tabla 4 muestran los datos encontrados a través de **Mininet** al momento de medir los parámetros delay y jitter para SDN en IpV6. Las mediciones que se realizaron para SDN IPv6 detallan valores bajos para los retrasos y las fluctuaciones que se querían medir en comparación a otro tipo de tecnologías de redes convencionales como OSPF y MPLS. Según (Sombredero & Silva, 2015) el protocolo de red OSPF, tiene Delay para 10, 100, 6000 y 30000 bytes los siguientes valores: 485.75, 520.57, 590.41 y 725.10 respectivamente. Para la prueba de Jitter los resultados encontrados por (Sombredero & Silva, 2015) fueron de 970.36, 1010.32, 1180.43 y 1792.75 respectivamente. Comparando estos resultados encontrados frente a los de SDN, se observan valores de errores muy bajos para las redes definidas por software frente a otros protocolos de red.

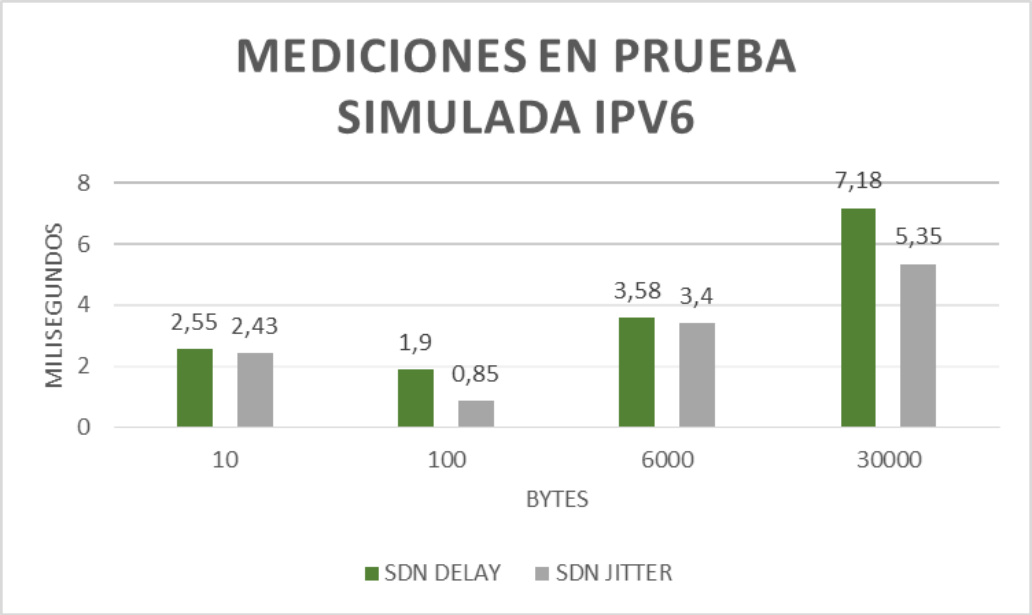


Figura 10: Gráfica comparativa medición de delay y jitter en SDN IpV6. Fuente propia

Para continuar realizando laboratorios y experiencias que permitan familiarizarse con esta nueva tecnología SDN y el emulador de red, se busca demostrar las ventajas de SDN frente a otros protocolos de red convencionales a través de una comparación de errores de delay y jitter entre el Multiprotocol Label Switching -Conmutación de Etiquetas Multiprotocolo- (MPLS) hallado por (Sombredero & Silva, 2015) y las redes definidas por Software SDN encontrado a través del comando ***show-protostad*** – opción ***verbase*** usado en **Mininet.** A continuación, se detallan los valores de delay encontrados para MPLS en paquetes de 10, 100, 6000 y 30000 bytes respectivamente: 4.10, 4.59, 7.18 y 12.55, todos valores en milisegundos. Para medir el jitter en esta tecnología convencional se tomaron la misma cantidad de paquetes de bytes y se hallaron los siguientes datos: 4.98, 5.49, 12.08 y 14.39 milisegundos.

IV. CONCLUSIONES

Mininet es un emulador que brinda condiciones óptimas para la simulación de las redes definidas por software, se destaca por ser una herramienta veloz en creación y en la ejecución de cualquier comando establecido dentro de la topología SDN, por lo tanto, es la clave para entender desde la práctica cómo funciona el protocolo OpenFlow. Esta guía servirá como referencia para futuras prácticas que involucren las redes definidas por software a través de **Mininet** y servirá como base para los acercamientos hacia el emulador de red y sus funciones. A partir de este momento la Universidad Santiago de Cali contará con una guía teórica y práctica que contenga los comandos principales para emular cualquier tipo de red y las herramientas básicas para que el interesado interactúe con estas nuevas formas de administrar las telecomunicaciones. La realización de los laboratorios sobre SDN trae consigo resultados positivos para el aprendizaje puesto que la práctica de automatización de redes es necesaria en la construcción del conocimiento sobre el área que se investiga. La interacción con **Mininet** reflejó las facilidades del emulador para crear entornos virtuales cercanos al deseado.

Mininet es una plataforma para la creación rápida de prototipos de red. Puede ejecutar un código de aplicación de red no modificado en redes pequeñas, así como en redes muy grandes. Es una alternativa para ejecutar experimentos SDN en redes emuladas. Los sistemas reales son complejos para reconfigurar. Las máquinas virtuales permiten cambios de topología más fáciles, pero sufren problemas de escalabilidad. Los simuladores son una buena alternativa, pero no se puede implementar el mismo código fuente en hardware real. Hay problemas de rendimiento en Mininet puesto que su

gran reto es el de modelar redes de gran escala con un rendimiento práctico.

Considerando los resultados encontrados durante este trabajo de grado y comparándolos frente a los hallazgos obtenidos por otros autores, es válido afirmar que la tecnología de red SDN tiene valores de retraso y fluctuaciones más bajos que otras tecnologías existentes. Según estos hallazgos el delay y jitter de las SDN son mucho más pequeños que las mismas mediciones en otros protocolos de red como OSPF y MPLS. Encontrar el valor de estos errores es importante puesto que permite tomar decisiones a la hora de implementar un protocolo de red en cualquier lugar. Así mismo, apoya lo ya planteando desde un inicio frente a las ventajas de aplicar SDN en una red.

REFERENCIAS

Bianco, A., Birke, L., & Palacin, M. (2010). Open Flow Switching: Data Plane Performance. En A. Bianco, L. Birke, & M. Palacin, *Open Flow Switching: Data Plane Performance.* Cape Town: 2010 IEEE International Conference on.

Cardona Aguirre, J. M., Melán, K., & Prado, H. F. (2016). *Tendencias, Mejoras y Robustamiento en redes SDN*. Cali: Universidad Santiago de Cali, clase de gestion de redes.

Casado, M. (2007). *Architectural support for security management in entreprise.* Stanford: Universidad de Stanford.

CISCO. (2005 de Abril de 2015). IT Certifications and Career Paths,. California.

Feamster, N., Rexford, J., & Zegura, E. (2013). The Road to SDN. *Queue - Large Scale Implementations Volumen 11*, 20.

Hata, H. (2013). A Study of Requirements for SDN Switch Platform. En H. Hata, *Intelligent Signal Processing and Communications Systems.* Naha: International Symposium communications.

Henao Ramirez, J. L. (2015). Informe Guía Teorico-Práctica Sobre Redes Definidas por Software para la Universidad Tecnologica de Pereira. *Universidad Tecnológica de Pereira*, 17-45.

HP. (14 de FEBRERO de 2015). *HP SDN App Store.* Recuperado el 20 de 01 de 2019, de HP SDN App Store: https://hpn.hpwsportal.com/catalog.html#/Home/Show.

ICESI, U. (01 de Enero de 2018). Redes Definidas en Software. *Contenido Programático*. Cali, Valle del Cauca, Colombia: Universidad ICESI.

Intel. (2015). Cómo evoluciona la infraestructura definida por software en Intel. *Documento Técnico*, 1-8.

Kreutz, D., Ramos, F., Vewrissimo, P., Rothenberg, E., Azodolmolky, S., & Uhlig, S. (2014). *Software-Defined Networking:A Comprehensive Survey.* New York: IEEE.

Llaudet Planas, A. (2003). Configuración de un entorno para emulación de SDN. *Universitat Politècnica de Catalunya*, 97.

MININET. (01 de Diciembre de 2014). *MININET*. Recuperado el 28 de Mayo de 2018, de http://Mininet.org/

Murillo Nogales, P. (2015). Análisis y evaluación de las redes definidas por software. *Universidad de Extrema Dura Merida*, 17.

Nadeau, T., & Gray, K. (2016). *SDN: Software Defined Networks, an authoritative review of networks programmability technologies.* Princeton: O`Reilly.

Nunes, B., Mendonca, M., Nguyen, X.-N., Obraczka, K., & Turletti, T. (2014). A survey of software-defined networking: Past, presents, and future of programmable networks. En B. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, & T. Turletti. Communications Surveys & Tutorials, IEEE.

Rao, N., Settlemyer, B. W., Liu, Q., Sen, S., Kettimuthu, R., Boley, J., . . . Yu, D. (2018). *Software-Defined Network Solutions for Science Scenarios: Performance.* Varanasi, India: International Conference on Distributed Computing and Networking,.

Risdianto, A. C., & Mulyana, E. (2012). Implementation and Analysis of Control and forwarding plane for SDN de

Telecommunication Systems, Services, and Applications (TSSA). En A. C. Risdianto, & E. Mulyana, *Implementation and Analysis of Control and forwarding plane for SDN de Telecommunication Systems, Services, and Applications (TSSA)*. Bali: 2012 7th International Conference on,.

Roncero, O. (21 de Enero de 2014). *Software Defined Networking*. Obtenido de Software Defined Networking: http://upcommons.upc.edu/pfc/bitstream/2099.1/21633/4/

Santos de Oliveira, R. L., Schweitz, M., Akira Shinoda, A., & Rodrigues Prete, L. (2014). *Using **Mininet** for Emulation and Prototyping Software-Defined Networks*. Sao Paolo: São Paulo State University "Julio de Mesquita Filho".

Sombredero, J. C., & Siva, F. (2015). Análisis comparativo entre redes SDN y redes convencionales. En J. C. Sombredero, & F. Siva, *Análisis comparativo entre redes SDN y redes convencionales* (pág. 124). Bogota: Universidad Santo Tomàs.

Stallings, W. (2015). *Foundations of Modern Networking SDN, NFV, QOE, IOT and CLOUD*. Indiana: Pearson.

Telecomunicaciones, O. I. (2015). *Informe Final Conferencia Mundial sobre las Telecomunicaciones 2015*. Dubai: OIT.

Velasco, R. (10 de Abril de 2014). *REDES ZONE*. Recuperado el 26 de Mayo de 2018, de http://www.redeszone.net/2014/03/20/lista-de-simuladores-de-redes-para-virtualizar-nuestra-propia-red/

Velasquez Vargas, W. (2014). *Emulacion de una red definida por software usando **Mininet***. Mexico.

Lopez, A. M., & Ramirez, M. (2018). Redes de datos definidas por software SDN. *Journal de Ciencia e Ingeniería*, 55-61.

Kaur, K., Singh, J., & Ghumman, N. S. (2018). Mininet as Software Defined Networking Testing Platform. *Shaheed Bhagat Singh State Technical Campus,*, 12-17.

COMO APORTE PARA LA UNIVERSIDAD SANTIAGO DE CALI, TAL Y COMO SE DESCRIBE EN ESTE ESCRITO DEJO UN ANEXO QUE SERVIRÁ COMO BASE PARA FUTURAS INVESTIGACIONES QUE PROMUEVAN EL CAMPO DE ESTUDIO PARA EL ÁREA DE INGENIERÍA ELECTRÓNICA, YA QUE CUENTA CON INFORMACIÓN RELEVANTE SOBRE LAS REDES PROGRAMABLES DE SOFTWARE Y LOS DEMÁS ASPECTOS A TENER EN CUENTA PARA EL ENTENDIMIENTO Y LA APLICACIÓN DEL MISMO.

Nota: Esta guía de instalación fue construida con base a prácticas sobre un sistema operativo Windows, pero no se limita al lector si usa un sistema operativo diferente. Se aclara que el emulador varía su configuración dependiendo de la práctica y la forma de ejecución.