UNIVERSIDAD
DE ANTIOQUIA

DISEÑO DEL COMPONENTE EXPERIMENTAL PARA LA FORMACIÓN EN REDES DE COMUNICACIONES DE NUEVA GENERACIÓN

Autor:

Juan Fernando Tamayo Galeano

Universidad de Antioquia

Facultad de Ingeniería

Departamento de Ingeniería Electrónica y Telecomunicaciones

Medellín, Colombia

2020

DISEÑO DEL COMPONENTE EXPERIMENTAL PARA LA FORMACIÓN EN REDES DE COMUNICACIONES DE NUEVA GENERACIÓN

JUAN FERNANDO TAMAYO GALEANO

Trabajo de grado presentado para para optar al título de:
Ingeniero de Telecomunicaciones

Asesor:

Ph.D. Juan Felipe Botero Vega

Universidad de Antioquia
Facultad de Ingeniería
Departamento de Ingeniería Electrónica y Telecomunicaciones
Medellín, Colombia
2020

Tabla de contenido

Resumen

En la actualidad las redes definidas por software (SDN) son un nuevo paradigma que conduce al cambio en cómo implementamos y administramos las redes. Al desprender el plano de control del plano de datos, se logra operar en un entorno centralizado, haciendo que las redes de comunicación sean programables, logrando mayor agilidad y flexibilidad al interior de una red.

En este proyecto se explican los conceptos básicos de SDN permitiendo, a través de la experimentación, compararlos con los escenarios tradicionales con las potencialidades que presentan las Redes Definidas por Software (SDN).

Se han desarrollado cuatro prácticas de laboratorio que explican los conceptos de Switching y Routing implementados a las redes de comunicaciones de nueva generación con enfoque en SDN (Software Defined Networking). En cada una de las guías de uso se explica cómo configurar cada una de las herramientas utilizadas y los procedimientos para desarrollar las prácticas, así como las pruebas para comprobar su correcto funcionamiento a través de una máquina virtual que tiene incorporado Mininet para la simulación de redes y POX como controlador OpenFlow.

Como escenario de prueba y evaluación de la plataforma remota diseñada, la Universidad de Antioquia implementará una plataforma educativa que permitirá apoyar procesos de formación en redes de comunicación. La plataforma integrará los contenidos teóricos del curso redes de nueva generación ofrecido por la Universidad de Antioquia, con un laboratorio remoto que permitirá el desarrollo de prácticas controladas desde un entorno virtual haciendo uso de contenedores a través de LXC OS.

1. Introducción

La innovación en las redes de comunicación continúa, y la necesidad de adaptarse a las limitaciones en las redes tradicionales, ha hecho que surjan nuevas tecnologías que contrarresten los costos operativos de una red.

Al día de hoy una red de computadores opera en un entorno distribuido, donde cada equipo al interior de una topología varía según su proveedor y es independiente a nivel de configuración, lo cual hace que se incremente la complejidad a la hora de gestionar equipos al interior de una red. [1] Las técnicas de conmutación y enrutamiento son un ejemplo de ello, dado que poseen un sistema robusto, el cual reduce la adaptabilidad de las redes en constante cambio.

Varios de los desafíos que enfrentan las redes tradicionales hoy en día, es poder realizar despliegues más rápidos, lograr interoperabilidad y la introducción de nuevos servicios, esto ha hecho que día a día surjan nuevas soluciones. Las SDN (Software Defined Networking) son consideradas como el nuevo modelo de arquitectura para suplir las problemáticas que traen las redes tradicionales, a través del funcionamiento de las redes de computadores en entornos programables se logra mayor flexibilidad, escalabilidad, abstracción y automatización de procesos de gestión al interior de una red.

Ante la gran evolución de las nuevas tecnologías e internet en todo el mundo, se ha globalizado los recursos informativos. La educación no se podía quedar atrás en el uso de las nuevas tecnologías por lo que ha ido evolucionando en función a ellas. El uso de las tecnologías de la información y la comunicación (TIC) como apoyo a los procesos formativos en la Universidad de Antioquia se ha convertido en una estrategia de alto impacto, logrando el cumplimiento sobre el compromiso social de brindar cobertura educativa para mejorar las oportunidades de las personas en las regiones y en el país.

Por medio del desarrollo de este proyecto el Departamento de Ingeniería Electrónica y Telecomunicaciones de la Universidad de Antioquia contará con material experimental para las asignaturas de redes de datos con enfoque SDN (Software Defined Networking) con el fin de contribuir a la formación en redes de comunicaciones de nueva generación.

Se llevarán a cabo varios casos prácticos orientados a comprender las generalidades y funcionamiento de una red SDN a través de la herramienta de

simulación de redes Mininet y el uso del controlador OpenFlow POX, los cuales estarán incorporados en una máquina virtual Ubuntu 16.04.3 la cual será el escenario de prueba y evaluación del correcto funcionamiento de cada uno de los escenarios. Además, se contará dentro de la máquina virtual con una guía de uso para cada caso práctico con su respectiva solución. De esta manera, se pretende ofrecer un entorno de laboratorio base sobre el cual permita experimentar el funcionamiento de las redes definidas por software y como futuro complemento a los contenidos teóricos de los cursos de redes de datos ofrecidos por la Universidad de Antioquia.

2. Objetivos

2.1. Objetivo General

Diseñar e implementar una plataforma experimental que se articule con los contenidos de los cursos teóricos de redes de la Universidad de Antioquia permitiendo a los estudiantes afianzar y ampliar su formación desde las redes tradicionales hacia paradigmas emergentes para redes de nueva generación como SDN.

2.2. Objetivos específicos

- Definir, a partir del contenido teórico a trabajar, los temas requeridos para implementar el componente experimental.

- Diseñar la plataforma experimental para la aplicación del contenido práctico de los cursos de redes de comunicación de la Universidad de Antioquia.

- Implementar la plataforma diseñada de forma que permita, a través de un entorno simulado, el desarrollo de las diferentes topologías de red, acorde a los contenidos teóricos de los cursos de redes de comunicación.

- Evaluar la funcionalidad de la plataforma implementada.

3. Marco Teórico

3.1. Arquitectura de red tradicional

Las redes de datos tradicionales se basan en dispositivos de red de función fija, como un conmutador o enrutador. Estos dispositivos tienen ciertas funciones que operan bien juntas y son compatibles con la red. Esta arquitectura se lleva a cabo en un entorno distribuido junto con un conjunto de protocolos definidos, los cuales hoy en día han constituido una tecnología fundamental que se ha optado para enviar y recibir datos a través de todo el mundo en los últimos años. [8]

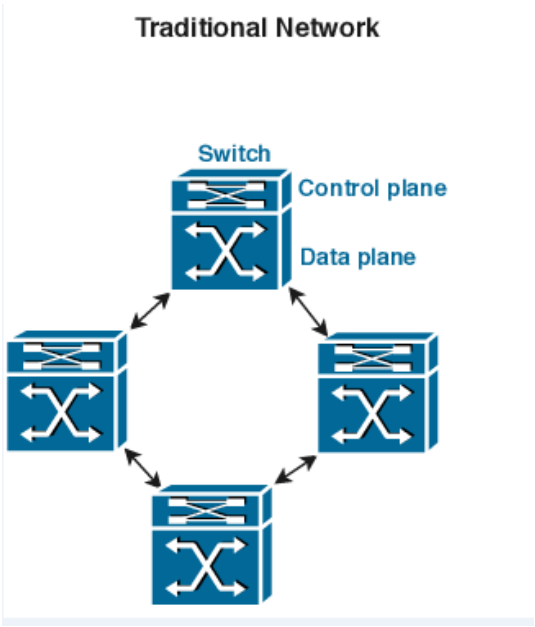

Figura 1. *Arquitectura de red tradicional. [8]*

3.1.1. Desventajas

Hoy en día la arquitectura de las redes tradicionales es muy estática, puesto que tanto el hardware como el software están integrados y limitados de una forma predeterminada. A partir de las necesidades dentro del mercado de las telecomunicaciones, las cuales están en constante evolución, en los últimos años se ha encontrado ante la necesidad de utilizar arquitecturas más flexibles, las cuales permitan obtener menor dependencia en los fabricantes, estándares

abiertos y dejar a un lado las soluciones aisladas que hoy ofrecen las redes tradicionales.

3.2. Switching

Los switches se usan para conectar varios dispositivos en la misma red. En una red diseñada correctamente, los switches LAN son responsables de controlar el flujo de datos en la capa de acceso y de dirigirlo a los recursos conectados en red.

Los switches Ethernet funcionan en la capa de enlace de datos, la capa 2, y se utilizan para reenviar tramas de Ethernet entre dispositivos dentro de una misma red. Sin embargo, cuando las direcciones IP de origen y destino están en distintas redes, la trama de Ethernet se debe enviar a un router. [2]

3.3. Routing

Los routers conectan una red a otra red. El router es responsable de la entrega de paquetes a través de distintas redes. El destino de un paquete IP puede ser un servidor web en otro país o un servidor de correo electrónico en la red de área local.

El enrutamiento es fundamental para cualquier red de datos, ya que transfiere información a través de una internetwork de origen a destino. Los routers son dispositivos que se encargan de transferir paquetes de una red a la siguiente.

Los routers descubren redes remotas de manera dinámica, mediante protocolos de routing, de manera manual, o por medio de rutas estáticas. En muchos casos, los routers utilizan una combinación de protocolos de routing dinámico y rutas estáticas. [2]

3.4. Arquitectura SDN

Las SDN (Software-Defined Networking) son un emergente paradigma que promete cambiar las limitantes de la actual infraestructura de redes. Se ha considerado como una de las soluciones más prometedoras para el futuro de Internet. SDN se caracteriza por sus dos características distinguidas, que incluyen el desacoplamiento del plano de control del plano de datos y la capacidad de programación para el desarrollo de aplicaciones de red. Como resultado, SDN está posicionada para proporcionar una configuración más eficiente, un mejor

rendimiento y una mayor flexibilidad para acomodar diseños de red innovadores. [3]

La arquitectura SDN está compuesta por tres capas, la capa de aplicación (plano de aplicación), la capa de control (plano de control) y la capa de infraestructura (plano de datos) que se muestran en la figura 2:

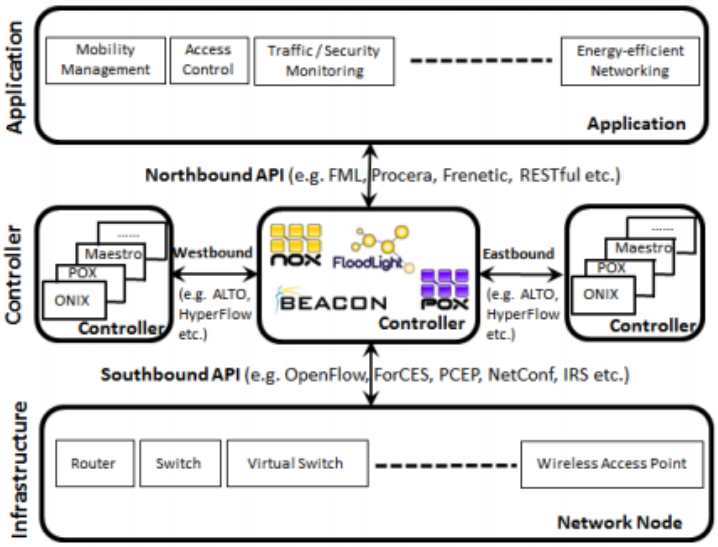

Figura 2. *Arquitectura funcional SDN que ilustra la infraestructura, el control y los elementos de aplicación de los que se compone la red. [6]*

3.4.1. Capa de Aplicación

Las aplicaciones SDN comunican sus requisitos a la red a través de una API que conecta con la capa de control, lo cual permite tener una visión global de las condiciones actuales de la red, con el fin de mejorar la transmisión de datos, mejorar la toma de decisiones y están diseñadas para satisfacer las necesidades de los usuarios. [4]

3.4.2. Capa de control

El controlador es el componente más importante de la arquitectura SDN ya que gestiona la capa de aplicación e infraestructura mediante dos interfaces, una con cada plano adjunto. Además, debe monitorizar todos los elementos para dar una visión global de la red.

Mediante la comunicación con el plano de infraestructura, se recoge el estado de la red y, según las exigencias de las aplicaciones, actualiza en los dispositivos las reglas de reenvío, ya que pueden producirse cambios como recuperación tras un fallo, migración de máquinas virtuales o balance de carga. También se debe mantener una validez y consistencia a fin de evitar bucles o agujeros de seguridad. [5]

3.4.3. Capa de Infraestructura

Está conformada por elementos de red, incluidos los conmutadores y enrutadores Ethernet. [6] Esto forma el plano de datos, los cuales son gestionados por medio del protocolo OpenFlow, permitiendo simplificar la configuración al administrador de red.

3.4.4. OpenFlow

OpenFlow es un protocolo que permite la creación de reglas en los dispositivos de red, programando un comportamiento deseado. En una red convencional, cada conmutador tiene software propietario que le dice qué hacer. Con OpenFlow se centralizan las decisiones de migración de paquetes, de modo que la red se puede programar independiente de los conmutadores individuales y equipo del centro de datos. [10]

3.4.4.1. Arquitectura

En el marco OpenFlow, el reenvío de paquetes se realiza a través del plano de datos y las decisiones de enrutamiento son tomadas desde el plano de control. Los conmutadores OpenFlow se encargan del reenvío de paquetes, mientras que un controlador configura las tablas de reenvío de conmutadores sobre una base de flujo, para permitir el aislamiento del flujo y la división de recursos. [10]

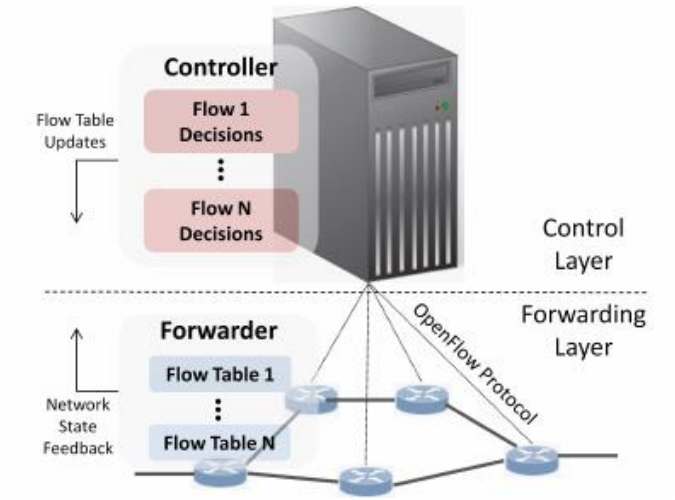

Figura 3. *Arquitectura OpenFlow. [9]*

3.4.4.2. Tablas de flujo

El modelo de encaminamiento usado por OpenFlow está basado en los flujos. Estos flujos se agrupan en tablas similares a las de enrutamiento y conmutación que de manera convencional ya se conocen. Las tablas de flujo son las encargadas de informar al conmutador que hacer con el reenvío del tráfico. [11] Estas tablas contienen los siguientes campos:

- **Match Fields:** Evalúa los campos del paquete.
- **Priority:** Define la prioridad de los paquetes.
- **Counters:** Almacena estadísticas en las tablas de flujo y actualiza los paquetes.
- **Instructions:** Indica las acciones a realizar con los paquetes.
- **Timeouts:** Tiempo de expiración de la tabla de flujo.
- **Cookie:** Este campo puede contener cualquier dato para el controlador.

3.4.4.3. Mensajes

El protocolo OpenFlow consta con tres tipos diferentes de mensajes. [12] Los mensajes que pueden ser enviados desde el switch y el controlador, los enviados por el switch y los enviados por el controlador.

- ***Mensajes del controlador al Switch:*** Son iniciados por el controlador, y su función es conocer y actuar sobre el estado del switch. Algunos de ellos son:

    - ***Modify-State:*** tiene como propósito agregar y eliminar entradas en las tablas de flujo.
    - ***Read-State:*** interroga al switch sobre estadísticas del tráfico.
    - ***Send-Packet:*** envía paquetes por un puerto específico en el switch.

- ***Mensajes de tipo simétrico:*** Son enviados sin solicitud en cualquier dirección. Entre los más importantes están:

    - ***Hello:*** mensajes intercambiados durante la negociación de una conexión.
    - ***Echo request/reply:*** mensajes intercambiados para conocer datos como latencia o el ancho de banda en cada extremo.

- ***Mensajes tipo asimétrico:*** Los switches envían mensajes al controlador a la llegada de un paquete, tras un error o tras un cambio de estado. Existen cuatro tipos:

    - ***Packet-in:*** enviado al controlador al recibir cualquier paquete que no tenga coincidencias en las tablas de flujo, o si la acción que corresponde a la entrada coincidente es reenviar el paquete al controlador.
    - ***Flow-removed:*** el conmutador notifica al controlador que una de sus entradas ha expirado.
    - ***Port-status:*** Informa cambio de estado en alguno de los puertos del switch.
    - ***Error.***

3.4.5. Controlador SDN

El componente principal de las redes definidas por software es el controlador. El controlador es lo que define la naturaleza del paradigma SDN. Es el componente responsable de concentrar la comunicación con todos los elementos programables de la red, proporcionando una vista unificada de la red. [13]

Mediante la comunicación con el plano de infraestructura, se recoge el estado de la red y, según las exigencias de las aplicaciones, actualiza en los dispositivos las reglas de reenvío, ya que pueden producirse cambios como recuperación tras un

fallo, migración de máquinas virtuales o balance de carga. También se debe mantener una validez y consistencia a fin de evitar bucles o agujeros de seguridad.

3.4.5.1. POX

Es un controlador de código abierto para desarrollar aplicaciones de control de SDN basadas en Python. Este se puede utilizar como un controlador SDN básico mediante el uso de los componentes que trae incluidos. Sin embargo, al ser de código abierto permite crear un controlador SDN más complejo mediante el desarrollo de aplicaciones de red que aborden la API de POX.

El controlador POX proporciona una manera eficiente de implementar el protocolo OpenFlow, que es el protocolo de comunicación de facto entre los controladores y los conmutadores. Con el controlador POX se pueden ejecutar diferentes aplicaciones de conmutación y enrutamiento. [14]

POX no es el único controlador de código abierto, existen desarrollos sobre otros lenguajes de programación para los controladores. La figura 4 muestra algunas características de estos controladores.

	Beacon	Floodlight	NOX	POX	Trema	Ryu	ODL
Soporte OpenFlow	OF v1.0	OF v1.0	OF v1.0	OF v1.0	OF v1.3	OF v1.0, v1.2, v1.3 y extensiones Nicira	OF v1.0
Virtualización	Mininet y Open vSwitch	Mininet y Open vSwitch	Mininet y Open vSwitch	Mininet y Open vSwitch	Construcción de una herramienta virtual de simulación	Mininet y Open vSwitch	Mininet y Open vSwitch
Lenguaje de desarrollo	Java	Java	C++	Python	Rudy/C	Python	Java
Provee REST API	No	Si	No	No	Si (Básica)	Si (Básica)	Si
Interfaz Gráfica	Web	Web	Python+, QT4	Python+, QT4, Web	No	Web	Web
Soporte de plataformas	Linux, Mac OS, Windows y Android para moviles	Linux, Mac OS, Windows	Linux	Linux, Mac OS, Windows	Linux	Linux	Linux, Mac OS, Windows
Soporte de OpenStack	No	Si	No	No	Si	Si	Si
Multiprocesos	Si	Si	Si	No	Si	No	Si
Código Abierto	Si	Si	Si	Si	Si	Si	Si
Tiempo en el mercado	4 años	2 años	6 años	1 años	2 años	1 años	5 meses
Documentación	Buena	Buena	Media	Pobre	Media	Media	Media

Figura 4. *Características de algunos controladores. [15]*

3.4.6. Network Slicing

Este paradigma de gestión de red tiene como idea esencial construir varias redes virtuales sobre una única red física compartida. Lo cual ha motivado a las empresas por adoptar el estándar SDN en sus redes. A través del aislamiento de las diferentes entidades que operan en una misma red, también brinda la capacidad de gestionar la red permitiendo habilitar y deshabilitar enlaces, realizar configuraciones de conectividad y ancho de banda entre subredes, permitiendo que cada entidad vea la red como si él fuera el único usuario. [16]

3.4.7. Herramientas de virtualización

3.4.7.1. Máquinas virtuales

Una máquina virtual es un software que al igual que un ordenador físico, ejecuta un sistema operativo y diferentes aplicaciones. Lo conforman un conjunto de archivos de configuración y especificación y cuenta con el respaldo de los recursos físicos de la máquina sobre la que se ejecuta. Cada una de estas máquinas virtuales tienen consigo un dispositivo virtual que proveen la misma funcionalidad que los dispositivos físicos. Entre los beneficios que traen consigo las máquinas virtuales son la portabilidad, manejabilidad y seguridad. [17]

3.4.7.2. Mininet

Mininet es un emulador de redes de código abierto a través del kernel de Linux. Hoy en día es una herramienta de apoyo para experimentar sistemas de SDN (Software Defined Networking) a través del protocolo OpenFlow. [18] Esta herramienta permite crear host, switches OpenFlow, controladores y enlaces, permitiendo generar diferentes topologías de red, incluidos desarrollos propios.

3.4.7.3. LXC OS

LXC es un contenedor dentro de una máquina host que ejecuta un sistema operativo completo aislado de dicha máquina. Desarrollar una virtualización del sistema operativo con LXC, implica la instalación de herramientas de espacio de usuario para implementar contenedores utilizando las funciones subyacentes del núcleo. [21]

3.5. Plataformas educativas

Actualmente se tienen plataformas MOOCs exitosas (e.g. Coursera y EdX) las cuales incluyen en su oferta cursos en redes de comunicación, estos contenidos son ofrecidos por institutos, organizaciones sin ánimo de lucro, corporaciones y organizaciones internacionales, algunos de estos contenidos son:

Curso	MOOC	Ofrecido por
Introduction to Cisco Networking	Coursera	Cisco
Fundamentals of Network Communication	Coursera	Sistema Universitario de Colorado
Packet Switching Networks and Algorithms	Coursera	Sistema Universitario de Colorado
Introduction to Open Source Networking Technologies	EdX	The Linux Foundation

Estos cursos llevan a cabo su parte experimental mediante simulación, lo cual permite ahorrar costos.

En la actualidad, EdX tiene plataformas compartidas con entidades colombianas, entre ellas está edX – Unalmed con la Universidad Nacional de Colombia, cedida por las universidades de Harvard y el MIT para el almacenamiento de información en forma de MOOCs, y alojada en un servidor Web administrado por Unalmed, quién distribuye y controla las actividades del diseño e implementación de los cursos virtuales (MOOCs), no presenciales (autogestión electrónica) elaborados por docentes que tienen experiencia en enseñar diferentes materias o cursos a nivel universitario.

Dentro de las principales funciones de la plataforma están; matricular usuarios, dar recursos, así como materiales y actividades de formación, permitir el acceso

mediante una contraseña, hacer seguimiento del proceso de aprendizaje con evaluaciones virtuales, realizar cuestionarios, gestionar la comunicación entre estudiante y profesor entre otros. [7]

4. Metodología

4.1. Temas requeridos

A partir del micro currículo Mooc Redes de Nueva Generación, se incorporaron los contenidos a desarrollar su componente práctico que permitan apropiar los conceptos sobre redes de comunicación de nueva generación con enfoque en SDN (Software Defined Networking).

Al abordar este contenido se identificaron tópicos como lo son: antecedentes y generalidades de las SDN, herramientas para la simulación de redes SDN, capa 2 y capa 3 en SDN. Siguiendo la temática anterior se estableció la siguiente estructura para el diseño del componente experimental que permita afianzar los conceptos que soportan el funcionamiento de las redes de comunicación en entornos programables:

1. Práctica Mininet
2. Práctica controlador SDN
3. Práctica Switching
4. Práctica Routing

A partir de la estructura anterior, se definió como elemento necesario para la apropiación de las prácticas contar con un escenario que permita desarrollar cada una de las actividades propuestas contenidas en las guías desde un entorno simulado.

4.2. Diseño e implementación del escenario de pruebas a través de un entorno simulado

Con el fin de contar con una plataforma experimental que permita la aplicación del contenido práctico definido, se optó por una máquina virtual con una versión de *Ubuntu 16.04.3* a través de VirtualBox, en la cual se instalaron las principales herramientas para el desarrollo de cada una de las prácticas propuestas. Entre estas se tiene Mininet para la simulación de redes, POX como herramienta para desarrollar la lógica de control, Sublime Text como editor de texto y Wireshark para captura de tráfico.

Los detalles generales de la máquina virtual que se incorporó, así como su interfaz de inicio se pueden visualizar en las figuras 5 y 6 respectivamente.

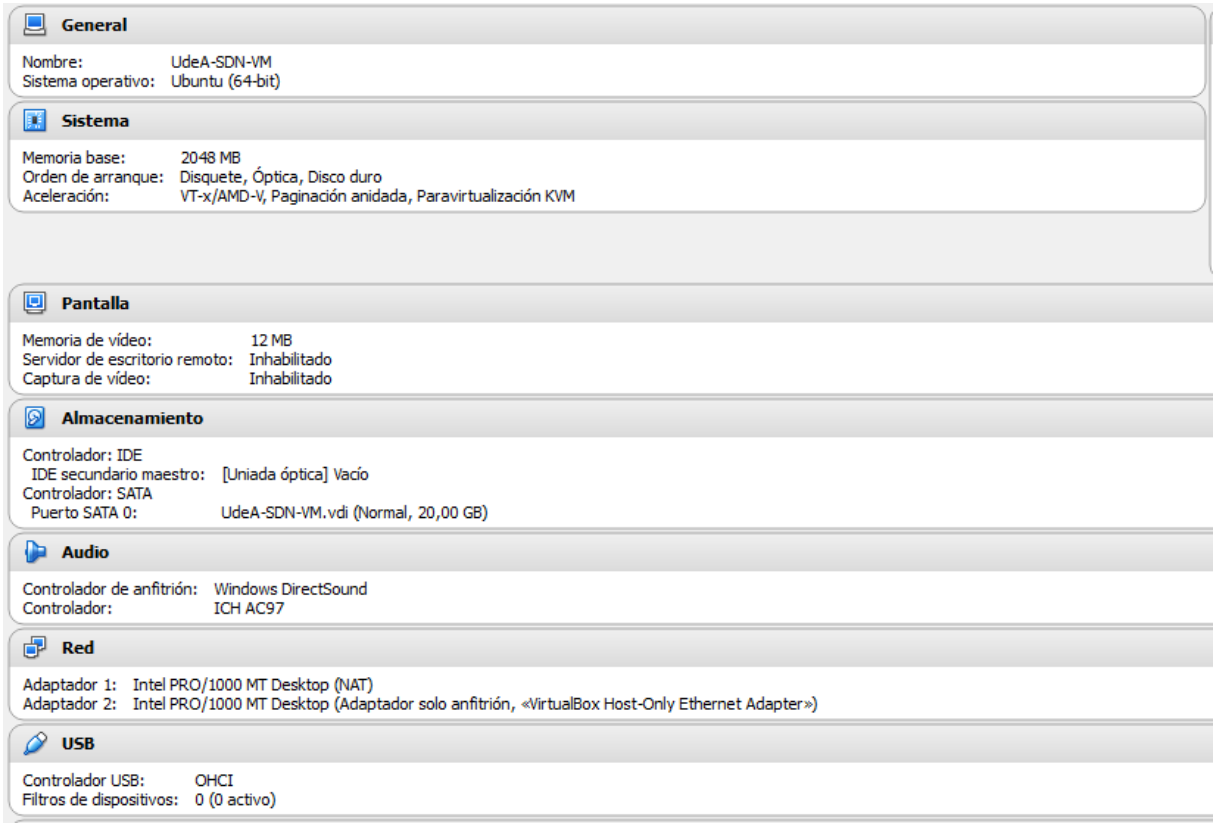

Figura 5. *Detalles del sistema*

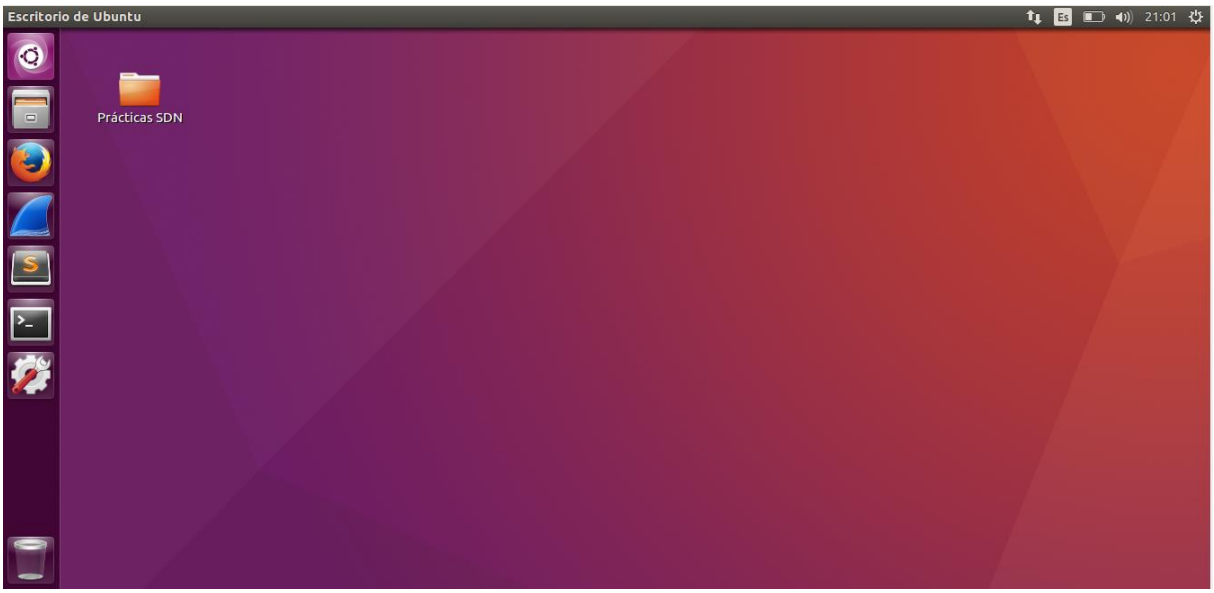

Figura 6. *Interfaz de inicio máquina virtual Ubuntu 16.04.3*

4.3. Entorno de trabajo

Desde la máquina virtual Ubuntu se cuenta de forma predeterminada con Python 2.7.12. La instalación de la herramienta Mininet se realizó de forma nativa descargando el código fuente desde el repositorio. A su vez esta forma de instalación trae consigo el controlador POX el cual fue utilizado para el desarrollo de las guías. La herramienta de edición de texto y la de captura de tráfico fueron descargadas e instaladas también de forma nativa.

Para el desarrollo del contenido en las guías se definieron ejercicios de ambientación que clarifican los conceptos teóricos definidos y familiarizan con las herramientas de simulación, finalizando con una actividad propuesta a desarrollar. Para cada una de las actividades propuestas en las guías se diseñó un formato con su solución, donde se brinda el desarrollo correspondiente y se muestra la evidencia del resultado obtenido.

5. Resultados y análisis

Se desarrollaron 4 prácticas sobre las cuales se aborda el manejo del emulador de redes SDN Mininet, el protocolo OpenFlow y se realizan desarrollos propios a través de Python sobre el controlador SDN POX. Adicional se desarrolló manual de virtualización del sistema operativo Ubuntu 16.04.3 haciendo uso de LXC OC.

Cada una de estas prácticas tienen como objetivo esencial ilustrar de forma experimental la teoría tratada acerca de las redes de nueva generación con enfoque SDN de manera que permita profundizar en temáticas de virtualización. Además, permiten evidenciar aquellas características que diferencian la tecnología de red tradicional con las SDN.

Las prácticas desarrolladas fueron las siguientes:

- **Guía Laboratorio 1: Uso de Mininet y Python**

Esta guía inicial está orientada al uso general del emulador Mininet. El objetivo consiste en la familiarización de los comandos básicos, diseño de topologías y uso de la API de Python, entre otros conceptos relacionados a la virtualización de redes.

Se divide en las siguientes secciones:

a. **Topologías y comandos básicos**

En esta sección se realiza una familiarización con los comandos típicos de Mininet, donde a través del diseño de una red básica se coloca a prueba el funcionamiento de la herramienta. Entre la lista de comandos básicos validados se tienen los siguiente:

- ***help:*** Visualizar listado de comandos de Mininet.
- ***nodes:*** Mostrar nodos de la topología iniciada.
- ***net:*** Mostrar los enlaces de la topología iniciada.
- ***dump:*** Ver información de todos los elementos de red de la topología iniciada.
- ***intfs:*** Visualizar interfaces de cada equipo de la red
- ***pingall:*** Probar conectividad en cada uno de los hosts que se tienen en la red.
- ***dpctl dump-flows:*** Visualizar tablas de flujo de los switches de la red. [19]

La salida en consola para la prueba de comandos básicos se puede visualizar en la figura 7.

```
UdeA-SDN-VM@t4m4y0-VirtualBox:~$ sudo mn
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> nodes
available nodes are:
c0 h1 h2 s1
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0
c0
mininet> links
h1-eth0<->s1-eth1 (OK OK)
h2-eth0<->s1-eth2 (OK OK)
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet>
```

Figura 7. *Prueba de comandos a través de una topología básica en Mininet.*

En esta sección también se explora el diseño de diferentes tipos de topologías predeterminadas que incluye Mininet, así como la variación en el tamaño de la red como la variación de enlaces. En la figura 8 se visualiza un ejemplo de variación de enlace a través de consola.

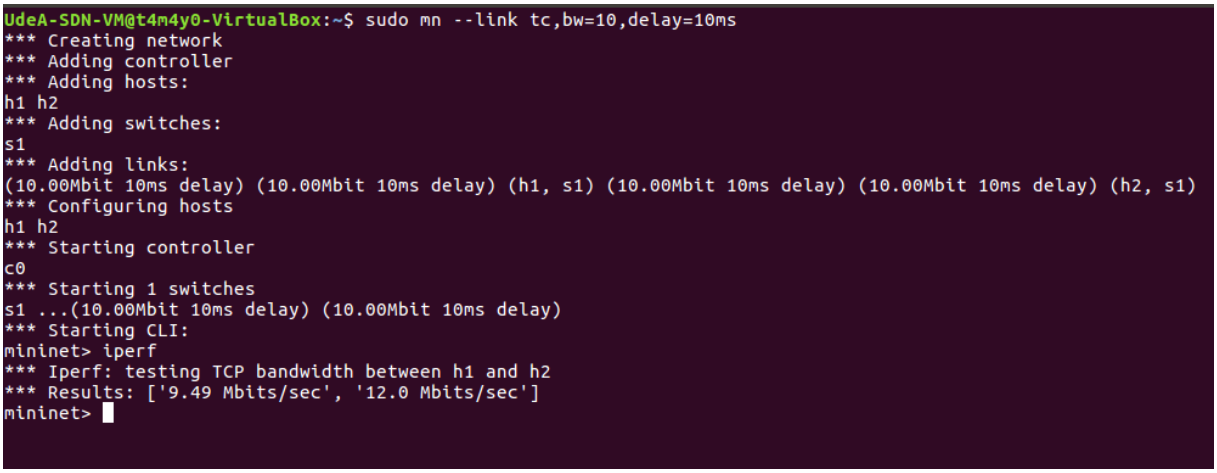

Figura 8. *Variación de enlace a través de consola.*

b. Uso de la API de Python

Para hacer un diseño más complejo sobre las topologías, se puede hacer uso de la API de Python, la cual permite crear una topología flexible que se puede configurar en función de los parámetros que se le ingrese y reutilizar para múltiples experimentos.

En esta sección se desarrolla la topología de la figura 9, y se explora sobre cada uno de los objetos y métodos que se utilizan a través de Python. Al contarse con el intérprete de texto ya incorporado sobre la máquina virtual, se facilita la programación de las topologías. La salida en consola se puede visualizar sobre la figura 10.

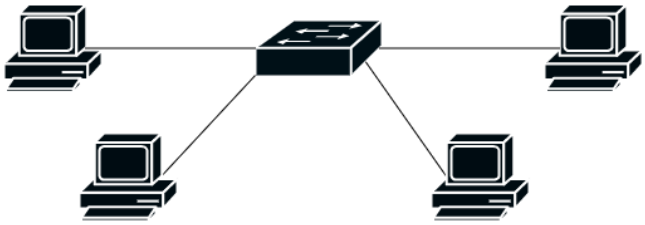

Figura 9. *Topología diseñada*

```
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
*** Stopping 1 controllers
c0
*** Stopping 4 links
....
*** Stopping 1 switches
s1
*** Stopping 4 hosts
h1 h2 h3 h4
*** Done
```

Figura 10. *Salida en consola del script de ambientación.*

c. Actividad propuesta

Como actividad final se plantea el desarrollo de la topología de la figura 11.

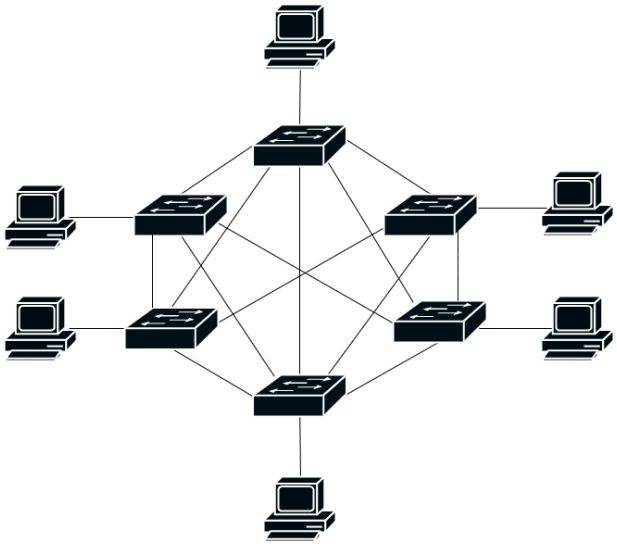

Figura 11. *Topología a diseñar.*

- **Guía Laboratorio 2: Uso del controlador SDN POX**

Esta guía está orientada al uso del controlador SDN POX, donde se familiariza con cada uno de los componentes predeterminados que trae, y se explora sobre la API de Python.

Se divide en las siguientes secciones:

a. Componentes

POX trae consigo diferentes componentes, los cuales no son más que programas en Python que implementan funcionalidades de red, estos se pueden invocar desde la línea de comandos cuando se inicia POX.

A continuación, se detallan algunos de los componentes. [20]

forwarding.hub: Configura una única regla de inundación, convirtiendo el swtich OpenFlow en un hub.

forwarding.l2_learning: Permite que los switches OpenFlow actúen como un switch de capa 2.

forwarding.l2_pairs: Este componente también hace que los switches actúen como un switch de capa 2, con la diferencia de que las reglas son basadas en las direcciones MAC.

forwarding.l3_learning: Permite examinar y elaborar solicitudes y respuestas ARP.

forwarding.l2_multi: Este componente permite el aprendizaje en base a una conexión switch a switch, como si cada switch tuviera solamente información local, para aprender la topología de toda la red. Tan pronto como un switch aprende dónde está una dirección MAC, todos lo hacen.

openflow.of_01: Este componente se comunca con switches OpenFlow 1.0. Este se inicia de forma predeterminada.

openflow.discovery: Este componente envía mensajes LLDP de conmutadores OpenFlow para que pueda descubrir la topología de la red.

openflow.spanning_tree: Utiliza el componente de descubrimiento para construir una vista de la topología de la red, construye un árbol de expansión, deshabilitando los puertos del switch que no están en el árbol permitiendo liberar bucles en las topologías.

info.packet_dump: Este componente permite descargar un ergistro con la información de los paquetes.

samples.pretty_log: Este componente brinda una salida de registro agradable y funcional en la consola.

proto.dns_spy: Este componente supervisa las respuestas de DNS y almacena sus resultados.

proto.dhcpd: Este es un servidor DHCP simple.

log.level: POX utiliza la infraestructura de registro de Python. Cada componente tiene sus propios registradores, cuyo nombre se muestra como parte del mensaje de registro. Este componente permite configurar qué registradores muestran qué nivel de detalle.

b. Aplicación

En esta sección se muestran diferentes ejemplos para ejecutar POX y explorar las tablas de flujo de los conmutadores OpenFlow. Logrando familiarizar con el uso del controlador POX a través de Mininet evidenciando como configurar y ejecutar diferentes componentes según las necesidades la red.

Como ejercicio de ambientación se plantea el uso del componente *forwarding.2_learning* sobre una topología en Mininet. Los resultados se visualizan en las figuras 12 y 13.

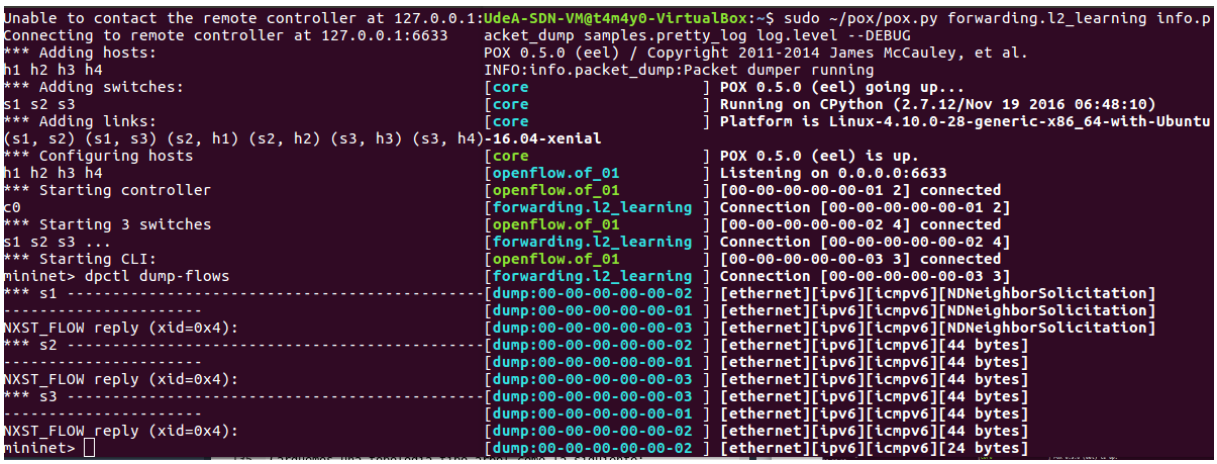

Figura 12. *Aplicación Mininet-POX parte 1.*

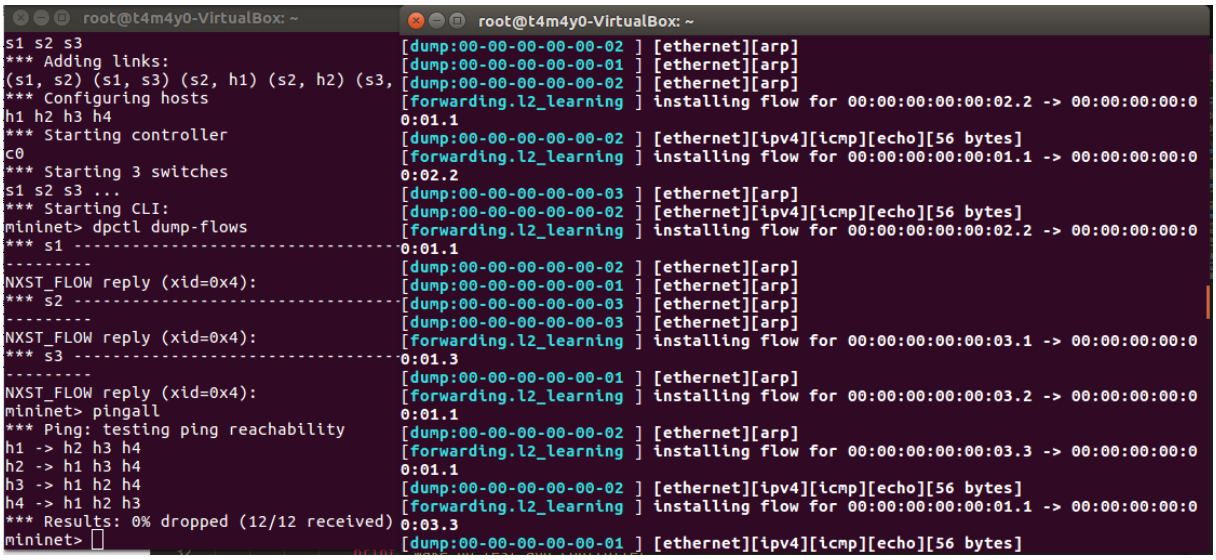

Figura 13. *Aplicación Mininet-POX parte 2.*

c. Actividad propuesta

Como actividad propuesta, se plantea diseñar la topología de la figura 14 y a su vez instalar reglas de flujo que evidencien el desarrollo a través de la API de POX.

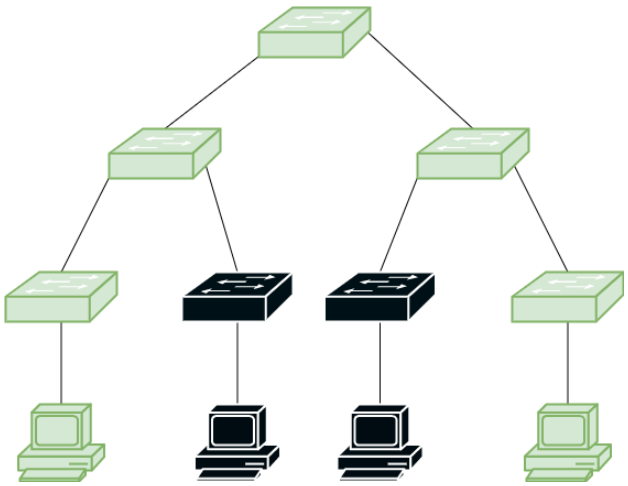

Figura 14. *Topología a diseñar.*

- **Guía Laboratorio 3: División de redes en entornos SDN**

Esta guía propone como dividir nuestra red SDN utilizando el controlador POX. El objetivo es mostrar alternativas en la virtualización de redes, en este caso a través del concepto Slicing SDN. En comparación a las redes tradicionales, un Slice puede asimilarse al concepto de VLAN, donde los dominios de broadcast quedan aislados permitiendo manipular el flujo del tráfico en tramos distintos de la red.

Se divide en las siguientes secciones:

a. Conocimientos previos

En esta sección haciendo uso del controlador POX se realiza un ejercicio de ambientación, donde a través del uso del *datapath-id* (*dpid*) se diferencia entre los conmutadores. El ejercicio desarrollado busca a partir de la topología de la figura 15, dividir la red como se visualiza en la figura 16.

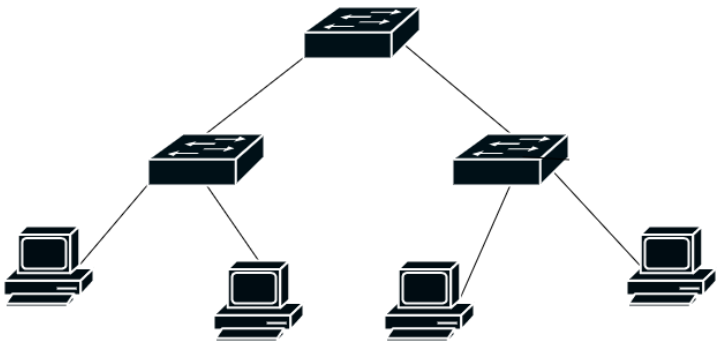

Figura 15. *Topología de ambientación*

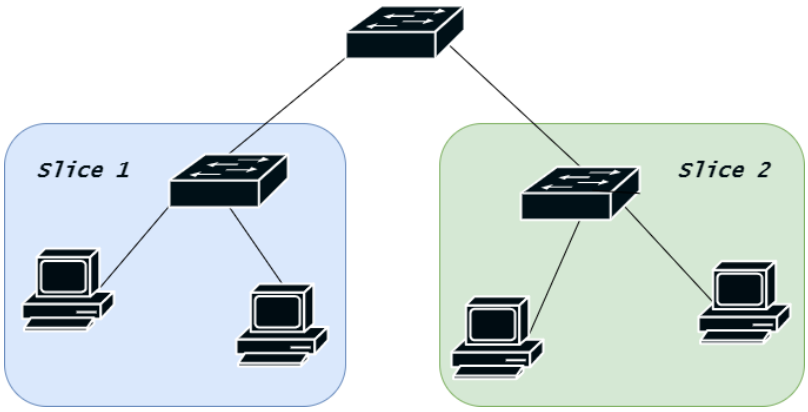

Figura 16. *Topología dividida*

b. Actividad propuesta

Para la actividad propuesta se entrega la topología de la figura 17 ya implementada en Mininet. A partir de esta se busca dividir en tres porciones la red haciendo uso del controlador POX.

Los slices finales se pueden visualizar en las figuras 18, 19 y 20.

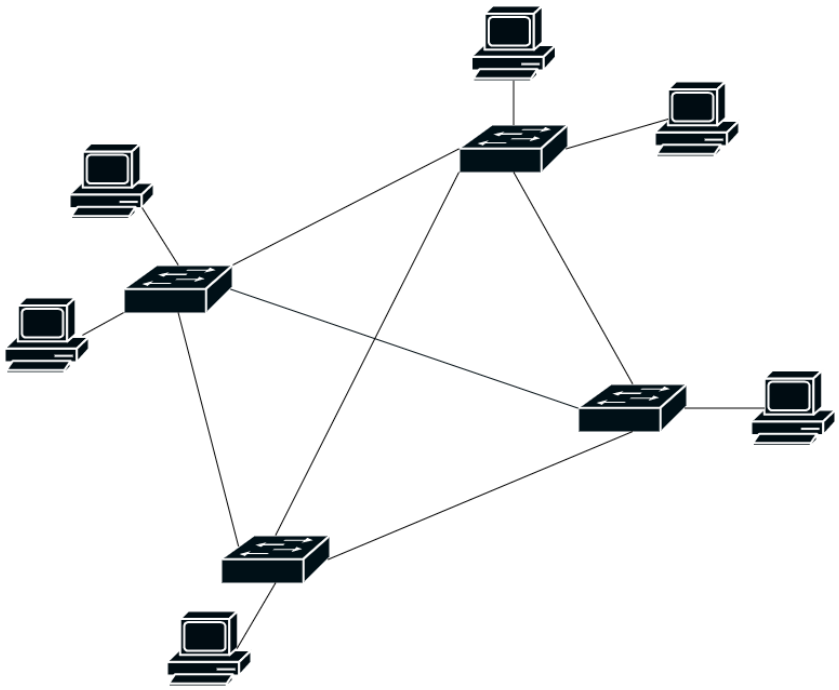

Figura 17. *Topología Actividad*

Figura 18. *Slice 1.*

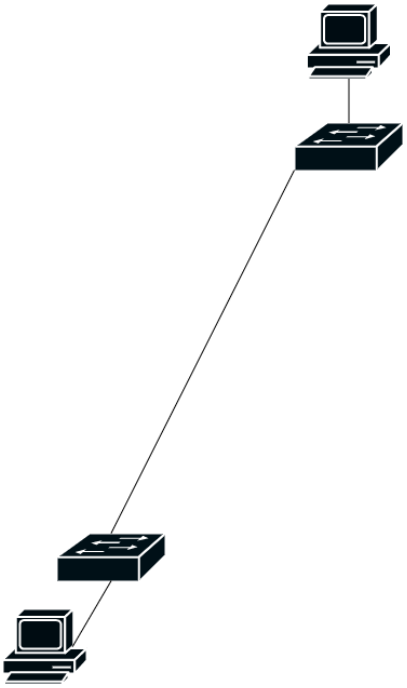

Figura 19. *Slice 2.*

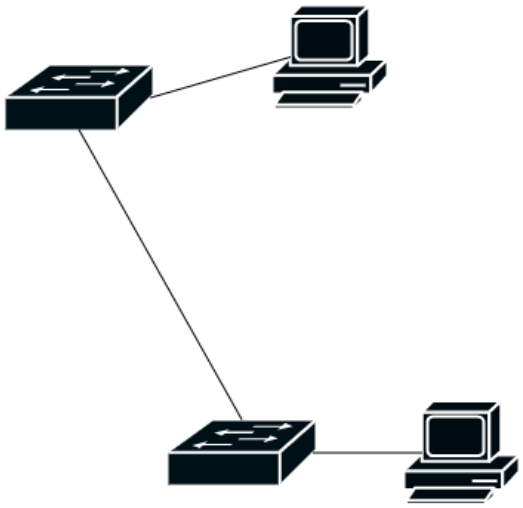

Figura 20. *Slice 3.*

- **Guía Laboratorio 4: Encontrar rutas en entornos SDN**

Esta actividad consiste en la posibilidad de definir el camino, de manera centralizada, que tomará un paquete cuyo destino es una dirección IP de una red específica. En este caso se hace uso de la API de python de mininet para generar una topología aleatoria de N nodos (switches) y al menos N-1 conexiones, garantizando que haya al menos un camino entre cada switch. Adicionalmente, cada switch "extremo", es decir, que sea fin de topología, tendrá un host conectado, como se observa en la figura 21. También se utilizará el módulo de detección de topología de POX, que permite programar el envío de mensajes LLDP (Link Layer Discovery Protocol) cuando se detecta activaciones o desactivaciones de enlaces. Dichos eventos tienen un atributo"enlace" que establece una correspondencia entre el switch que envía el evento y el puerto físico asociado al enlace. Así, se puede conocer cómo se conectan los switches de la topología:

```
INFO:openflow.discovery:link detected: 00-00-00-00-00-05.3 -> 00-00-00-00-00-0
2
-----switch1 se conecta con:-----
switch4 en el puerto4
switch6 en el puerto2
switch3 en el puerto3
-----switch2 se conecta con:-----
switch6 en el puerto4
switch4 en el puerto5
switch5 en el puerto2
switch3 en el puerto3
-----switch3 se conecta con:-----
switch6 en el puerto4
switch2 en el puerto2
switch4 en el puerto5
switch1 en el puerto3
-----switch4 se conecta con:-----
switch2 en el puerto4
switch1 en el puerto6
switch6 en el puerto2
switch3 en el puerto5
switch5 en el puerto3
-----switch5 se conecta con:-----
```

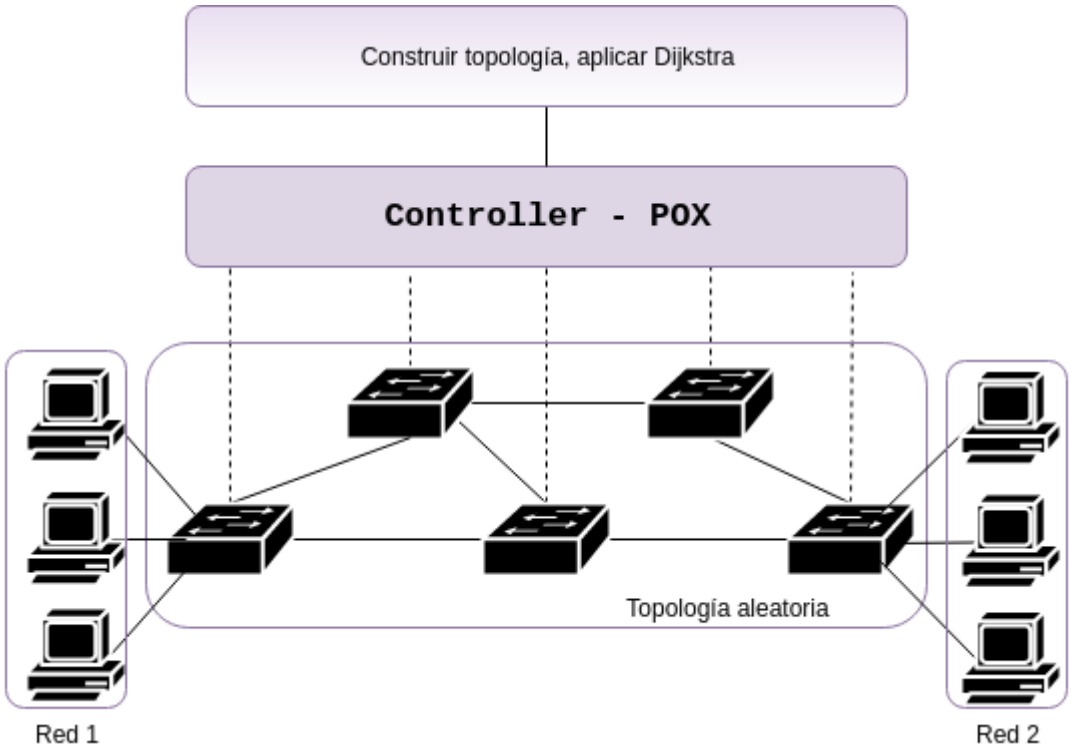

Figura 21. *Topología aleatoria*

a. Conocimientos previos:

El estudiante, conociendo la topología, debe instalar las reglas adecuadas en los switches para garantizar conectividad completa, esto lo puede hacer instalando directamente las reglas en OVS o programar el envío desde POX de los mensajes de OF necesarios. En este caso, la aplicación (si se desarrolla) no reacciona ante

cambios en la topología, como por ejemplo la eliminación o adición de enlaces. Esto muestra una clara ejecución de un escenario de enrutamiento estático, donde un administrador (en este caso desde POX) instala las rutas en cada uno de los equipos de red utilizando un criterio arbitrario.

b. Actividad propuesta:

Con el fin de aproximarse al enrutamiento dinámico, en este caso las rutas serán calculadas por un agente centralizado (POX) utilizando un algoritmo que, con una métrica definida, puede hallar los caminos más cortos de un origen a un destino. Para esto se programó el algoritmo de dijkstra [22] en una función que recibe el grafo asociado a la topología de red y calcula los caminos más cortos a todos los destinos desde un nodo fuente. El algoritmo entregará el camino completo hacia una red particular, como se observa en la figura 22.

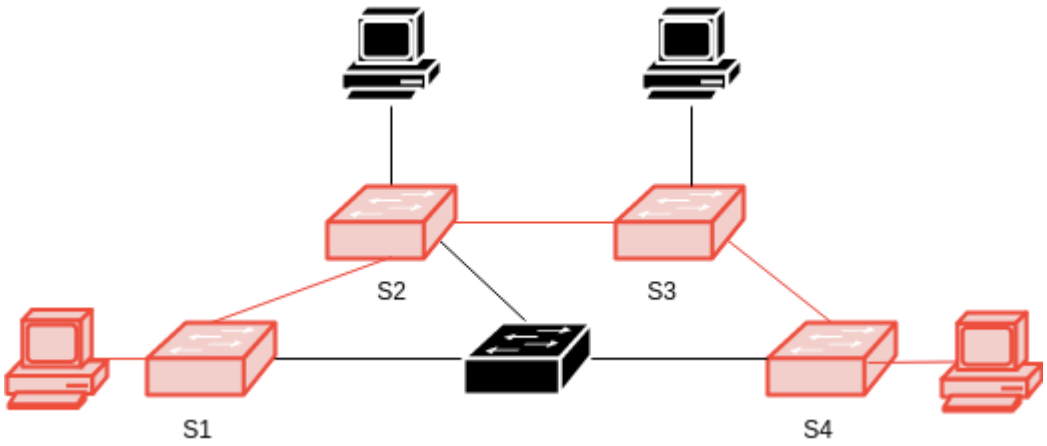

Figura 22. *Ruta entregada por el algoritmo*

En este caso, el estudiante debe encontrar la manera correcta de enviar la topología a la función de dijkstra y traducir la información que devuelve la función. A la función se le debe enviar un nodo fuente y la topología (que se guarda en un diccionario de python) para que calcule las rutas a todos los destinos. Dicha función entrega un grafo, que normalmente es un spanning tree, del cual el estudiante debe derivar las reglas para ser instaladas a través de OF. Se debe verificar conectividad completa entre los hosts de la topología.

Adicionalmente, se debe manejar también los cambios en los enlaces de la topología y recalcular las rutas ante estos eventos. Esto se logra al aprovechar la funcional de descubrimiento de topología de POX y ejecutar Dijkstra al recibir un evento de link encendido o apagado.

- **Manual virtualización LXC OS**

Este manual muestra el paso a paso de cómo crear un contenedor a través de LXC, y a su vez la puesta en práctica haciendo uso de Mininet.

Está dividido en las siguientes secciones:

a. Instalación LXC

En esta sección se proporcionan los comandos para instalar las herramientas de espacio de usuario de LXC junto con la librería de ayuda *libvirt*, la cual funciona con muchas tecnologías de virtualización y proporciona una interfaz común para interactuar con estas.

La verificación de las herramientas de espacio de usuario para LXC que permiten el correcto funcionamiento en el sistema operativo host, se visualizan en la figura 23.

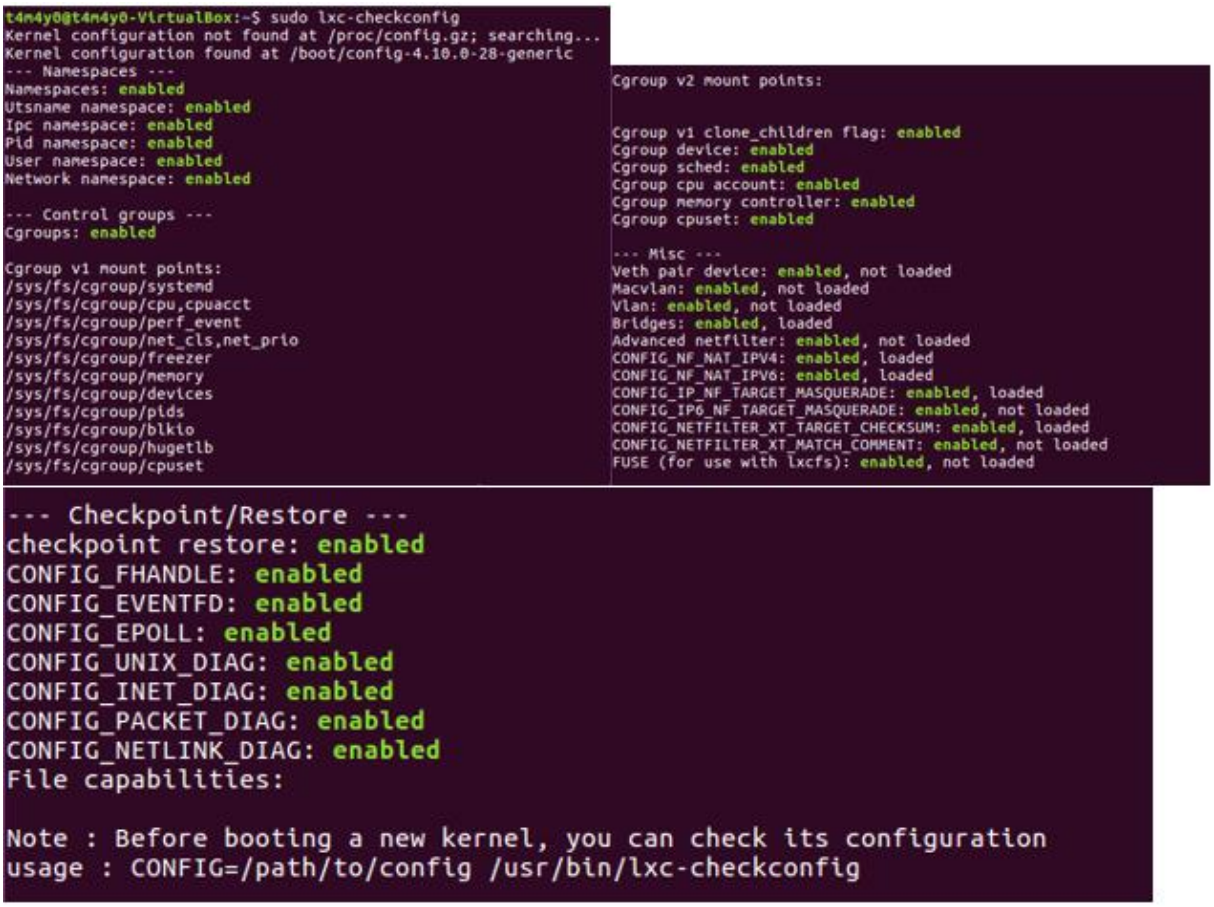

Figura 23. *Salida del comando lxc-checkconfig para validación del correcto funcionamiento para LXC.*

b. Virtualización SO Ubuntu

En esta sección se desarrolla la virtualización sobre la máquina host Ubuntu. Incluyendo el paso a paso para la definición del contenedor, así como su almacenamiento, configuración e inicialización de este.

Una vez completado el proceso de inicialización en consola del contenedor, se desarrolla la instalación de Mininet y POX. Las salidas en consola más destacadas se pueden visualizar en las figuras 24, 25, 26 y 27.

```
t4m4y0@t4m4y0-VirtualBox:~$ sudo virsh -c lxc:// dumpxml prueba
<domain type='lxc'>
  <name>prueba</name>
  <uuid>192a17a5-f4c9-433c-b0cc-79bfa3984ec3</uuid>
  <memory unit='KiB'>1048576</memory>
  <currentMemory unit='KiB'>1048576</currentMemory>
  <vcpu placement='static'>1</vcpu>
  <os>
    <type arch='x86_64'>exe</type>
    <init>/sbin/init</init>
  </os>
  <clock offset='utc'/>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>destroy</on_crash>
  <devices>
    <emulator>/usr/lib/libvirt/libvirt_lxc</emulator>
    <filesystem type='mount' accessmode='passthrough'>
      <source dir='/var/lib/lxc/prueba/rootfs'/>
      <target dir='/'/>
    </filesystem>
    <interface type='network'>
      <mac address='52:54:00:eb:97:09'/>
      <source network='default'/>
    </interface>
    <console type='pty'>
      <target type='lxc' port='0'/>
    </console>
  </devices>
</domain>
```

Figura 24. *Almacenamiento y configuración del contenedor.*

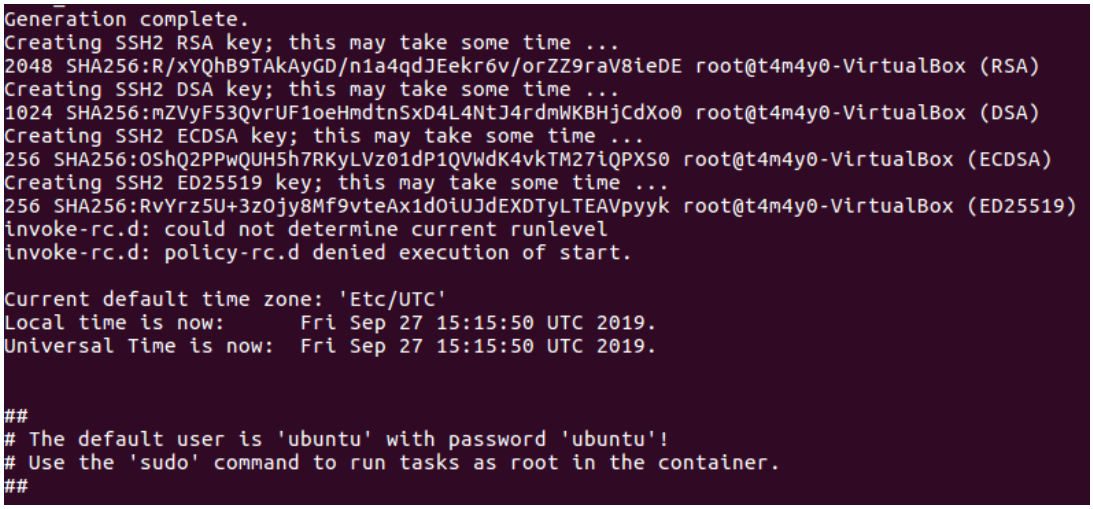

Figura 25. *Salida del proceso de creación del contenedor.*

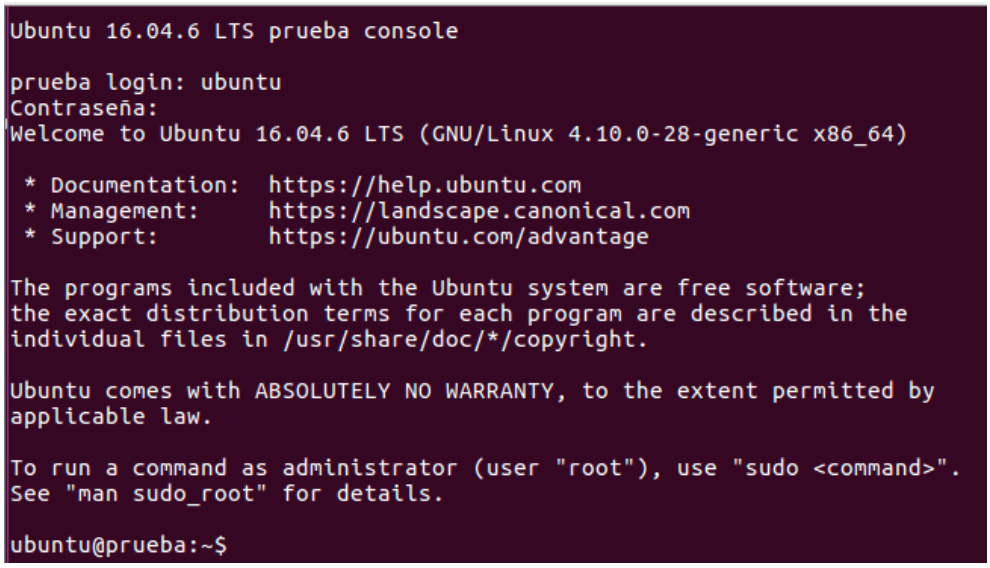

Figura 26. *Autenticación y acceso a la terminal del contenedor de prueba*

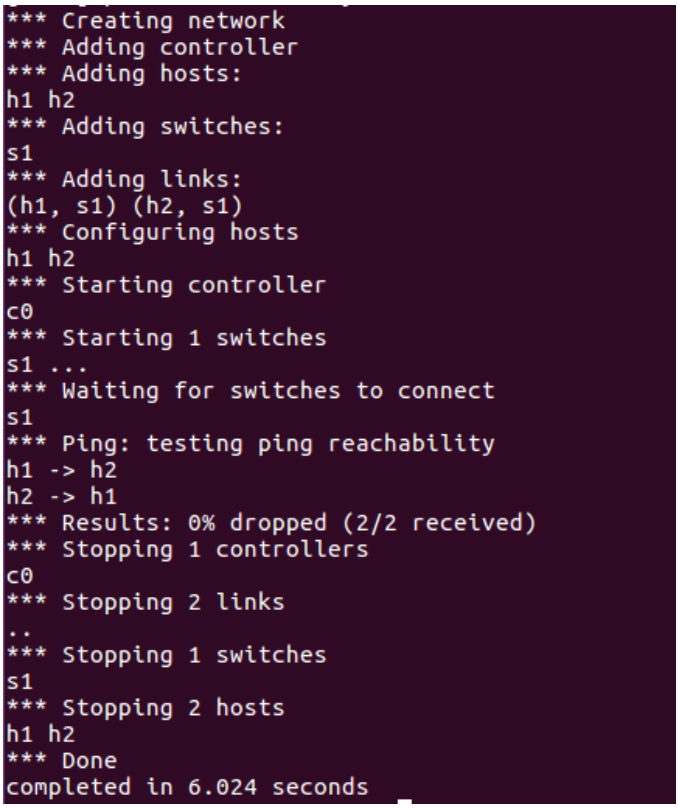

Figura 27. *Salida prueba Mininet.*

6. Conclusiones

Se entrega una plataforma funcional, que contiene el componente experimental necesario para un curso introductorio de redes definidas por software. Las características de dichas plataformas se pueden extender a cursos masivos en línea, gracias a la posibilidad de utilizar técnicas de virtualización como LXC.

A lo largo del presente trabajo se ha puesto a prueba las potencialidades que traen consigo las Redes Definidas por Software. A través de un entorno simulado se ha descrito la transición desde las redes activas hacia paradigmas emergentes, poniendo a prueba una de las principales características de las SDN: la programabilidad.

Con el desarrollo de este trabajo se genera material clave para la familiarización en la herramienta Mininet y uso del controlador POX lo cual facilitará los procesos formativos en temas de virtualización y redes de comunicación de nueva generación con enfoque en SDN.

Finalmente, como línea futura al trabajo realizado, se espera que sobre cada una de las simulaciones realizadas se lleve a cabo la implementación correspondiente

en equipos reales, permitiendo de esta manera que los usuarios estén en capacidad de adoptar SDN en el ámbito laboral.

Referencias Bibliográficas

[1] Nick Feamster, Jennifer Rexford, and Ellen Zegura, "The Road to SDN: An Intellectual History of Programmable Networks", ACM SIGCOMM CCR, April 2014 http://www.cs.princeton.edu/courses/archive/fall13/cos597E/papers/sdnhistory.pdf

[2] Cisco Networking Academy. Fundamentos de enrutamiento y conmutación (Routing and Switching Essentials). CCNA2 V5. https://juliorestrepo.files.wordpress.com/2015/03/pdf_ccna2_v5.pdf

[3] Xia, W.; Weng, Y.; Foh, C. H.; Niyato, D.; Xie, H. (2015) "A Survey on Software-Defined-
Networking" http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6834762

[4] Maldonado Hidalgo, Diego Armando. (2014). Diseño e implementación de una aplicación de red bajo la arquitectura SDN. PONTIFICIA UNIVERSIDAD JAVERIANA https://repository.javeriana.edu.co/bitstream/handle/10554/12745/MaldonadoHidalgoDiegoArmando2014.pdf?sequence=1&isAllowed=y

[5] Serrano Carrera, David Andrés. (2015). Trabajo fin de grado. Redes Definidas por Software (SDN): OpenFlow. Universitat politécnica de Valencia. https://riunet.upv.es/bitstream/handle/10251/62801/SERRANO%20-%20Redes%20Definidas%20por%20Software%20(SDN):%20OpenFlow.pdf?sequence=3

[6] S. Sezer, S. Scott-Hayward, p. K. Chouhand et al," Are we Ready for SDN? Implementation Challenges for Software-Defined-Networks", in IEEE Communications Magazine, vol. 51, pp. 36-43, 2013

[7] Quiroz Tobón, Hernán. (2016). Creación y diseño de un curso MOOC-Cálculo Integral en la plataforma edX-Unalmed. Universidad Nacional de Colombia http://bdigital.unal.edu.co/52974/1/8404948.2016.pdf

[8] Malik, Ali & Aziz, Benjamin & Ke, Chih-Heng. (2018). THRIFTY: Towards High Reduction In Flow Table memorY. 10.4230/OASIcs.ICCSW.2018.2.

[9] Bianco, Andrea & Birke, Robert & Giraudo, Luca & Palacin, Manuel. (2010). OpenFlow Switching: Data Plane Performance. IEEE International Conference on Communications (ICC). 1-5. 10.1109/ICC.2010.5502016.

[10] Open Networking Foundation. (2015). OpenFlow Switch Specification (Version 1.5.1). Current.

[11] Qian, Ying & You, Wanqing & Qian, Kai. (2016). OpenFlow flow table overflow attacks and countermeasures. 205-209. 10.1109/EuCNC.2016.7561033.

[12] Klein, Dominik & Jarschel, Michael. (2013). An OpenFlow extension for the OMNeT++ INET framework. 322-329. 10.4108/simutools.2013.251722.

[13] Prete, Ligia & Shinoda, Ailton & Schweitzer, Christiane & Oliveira, Rogerio. (2014). Simulation in an SDN network scenario using the POX Controller. 1-6. 10.1109/ColComCon.2014.6860403.

[14] Kaur, Sukhveer & Singh, Japinder & Ghumman, Navtej. (2014). Network Programmability Using POX Controller. 10.13140/RG.2.1.1950.6961.

[15] Centeno, Alejandro & Rodriguez Vergel, Carlos Manuel & Anías Calderón, Caridad & Camilo, Frank & Bondarenko, Casmartiño & Uci,. (2014). Controladores SDN, elementos para su selección y evaluación. Telemática. 13. 10-20.

[16] Costanzo, Salvatore & Fajjari, Ilhem & Aitsaadi, Nadjib & Langar, Rami. (2018). A Network Slicing Prototype for a Flexible Cloud Radio Access Network. 10.1109/CCNC.2018.8319259.

[17] VMware, What Is a Virtual Machine? [online] Available at: https://pubs.vmware.com/vsphere-51/index.jsp?topic=%2Fcom.vmware.vsphere.vm_admin.doc%2FGUID-CEFF6D89-8C19-4143-8C26-4B6D6734D2CB.html [Accessed 25 Dic.2019].

[18] Oliveira, Rogerio & Schweitzer, Christiane & Shinoda, Ailton & Prete, Ligia. (2014). Using Mininet for emulation and prototyping Software-Defined Networks. 2014 IEEE Colombian Conference on Communications and Computing, COLCOM 2014 - Conference Proceedings. 1-6. 10.1109/ColComCon.2014.6860404.

[19] Mininet Walkthrough [online] Available at: http://mininet.org/walkthrough/ [Accessed 3 Jan.2020]

[20] POX Wiki [online] Available at: https://openflow.stanford.edu/display/ONL/POX+Wiki.html [Accessed 3 Jan.2020]

[21] S., Senthil. (2017). Practical LXC and LXD: Linux Containers for Virtualization and Orchestration. 10.1007/978-1-4842-3024-4.

[22] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. Numerische Mathematik, 1(1), 269–271.