PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE INGENIERÍA

MAESTRÍA EN REDES DE COMUNICACIÓN

INFORME FINAL CASO DE ESTUDIO PARA UNIDAD DE TITULACIÓN ESPECIAL

PREVIO LA OBTENCION DEL TÍTULO DE:

MASTER EN REDES DE COMUNICACIÓN

TEMA:

"DISEÑO E IMPLEMENTACIÓN DE UN FIREWALL L2 UTILIZANDO REDES DEFINIDAS POR SOFTWARE (SDN)"

JAIME ADRIÁN BONILLA FERNÁNDEZ

Quito – 2016

AUTORÍA

Yo, Jaime **Adrián Bonilla Fernández**, portador de la cédula de ciudadanía No.**0201532249**, declaro bajo juramento que la presente investigación es de total responsabilidad del autor, y que se he respetado las diferentes fuentes de información realizando las citas correspondientes. Esta investigación no contiene plagio alguno y es resultado de un trabajo serio desarrollado en su totalidad por mi persona.

Jaime Adrián Bonilla Fernández

Contenido

1. Introducción

Las empresas cada día son más dependientes de las redes informáticas, por tal razón los firewall son un aspecto primordial para que la continuidad de las operaciones de la empresa no se vean comprometidas por cualquier amenaza de seguridad por mínima que ésta sea. Como las Redes Definidas por Software (SDN) está teniendo gran acogida en la sustitución de la arquitectura tradicional, debido a que éstas hacen mucho más fácil el extender y desarrollar nuevas funcionalidades, a través de aplicaciones que se ejecutan en un controlador; sería muy interesante conocer las características de seguridad que puede ser proporcionada por esta nueva tecnología.

Este proyecto que presentará una estrategia para la implementación de un Firewall SDN, apoyado en un emulador de redes como lo es Mininet, se ha dividido en tres grandes bloques para su desarrollo.

En la primera fase se realizará el estudio teórico de las redes SDN, lo cual ayudará a tener un visión completamente clara de cómo funciona dicha tecnología, así como también se analizará la utilidad de la herramienta Mininet en la creación de topologías de red (controlador SDN, OpenFlow Switches, etc.) que apoyarán al entendimiento de las Redes Definidas por Software.

La segunda fase es propiamente la del diseño del Firewall, en donde la parte medular de la misma se enfocará en el desarrollo de un módulo para el controlador SDN, el cual en conjunto

con el protocolo Open Flow será quien tome las decisión de si un paquete de datos, debe ser bloqueado o permitido según las políticas de seguridad que se defina para el manejo del tráfico.

Finalmente se realizará una emulación de la red mediante la utilización del programa MiniNet, con lo cual se tendrá la posibilidad de experimentar con este nuevo tipo de redes, y demostrar como la gestión y el control del flujo de datos en una red es mucho más viable con la tecnología SDN.

2. Justificación

Aunque las empresas reconocen que la seguridad de la información es un tema que debe ser prioritario, no es de sorprenderse que esto no sea una realidad; exponiéndose de esta manera a que cualquier pequeña fuga de información conlleve a que su negocio desaparezca.

Uno de los componentes de seguridad que se requieren dentro de una organización es el firewall, un elemento que permite controlar el tráfico de red tanto hacia fuera como dentro de la misma, sin embargo debido al costo de estos equipos o a la complejidad de programación de los mismos, muchas empresas optan por obviar este elemento importante, dejando de esta manera expuesta su información a múltiples amenazas de seguridad.

Por lo mencionado, las empresas deben pensar en estrategias que les permitan garantizar la accesibilidad y seguridad de la información, y es en este contexto es de interés estudiar y aplicar la tecnología SDN cuya principal motivación es la reducción de los costes en equipamiento y operación de red.

Gracias a esta tecnología se podrá potencialmente reducir la complejidad de las arquitecturas de seguridad así como también se tendrá la posibilidad de solicitar recursos de red en tiempo real, como ancho de banda, calidad de servicio, rutas de protección, etc.; en definitiva con SDN, la red actual, muy estática, puede evolucionar en una plataforma de despliegue de aplicaciones y servicios, capaz de responder rápidamente a los cambios en las necesidades del negocio y del usuario final.

3. Antecedentes

El término SDN (Software Defined Network o Red Definida por Software), hace referencia a una arquitectura de red que trata de la separación efectiva de los planos de control y de datos, con lo cual la gestión de las redes ya no dependería de una solución de hardware, sino de un controlador de software más inteligente y preparado para exprimir el rendimiento al máximo.

En otras palabras SDN es una manera de abordar la creación de una red en la cual el control se desprende del hardware y se le da el mismo a una aplicación de software llamada controlador, con lo cual se consigue redes más programables, automatizables y flexibles.

Una de las formas más populares de implementar SDN es el protocolo OpenFlow, que es soportado por la ONF (Open Networking Foundation). OpenFlow es un estándar abierto que busca la interoperabilidad entre distintos fabricantes; el código abierto es considerado por muchos operadores como la alternativa más rápida para conseguir soluciones interoperables o multivendedor, respecto a la estandarización de las API, que requerirá de más tiempo.

Para obtener el máximo rendimiento, utilización y simplicidad, las operadoras de telecomunicaciones, proveedores de servicios en la nube y las empresas, pueden utilizar la tecnología SDN en toda la red, desde el centro de datos hasta el escritorio, independizando la infraestructura hardware del plano de control, y de las aplicaciones; ello permitirá solucionar las complejidades existentes hoy en día y mejorar la agilidad del negocio. Para dar respuesta a los

desafíos que han generado las redes convencionales, las organizaciones necesitan ser capaces de automatizar la red de extremo a extremo y SDN hace posible independizar el plano de control de la infraestructura física.

4. Objetivos

Objetivo General:

Configurar un entorno que permita diseñar e implementar un firewall de capa 2 utilizando SDN (Redes Definidas por Software) con el fin verificar los beneficios que esta tecnología aportaría en temas de seguridad de la información de una red.

Objetivos Específicos:

1. Conocer las características más importantes de las redes definidas por software (funcionamiento, arquitectura, otros).
2. Implementar la topología de la red, mediante el uso de un software de virtualización (o emulación) de redes llamado Mininet.
3. Administrar el plano de control de los dispositivos, mediante un controlador de SDN, responsable de gestionar todo lo referente al comportamiento de la red
4. Analizar el comportamiento de la red antes y después de aplicar políticas de firewall.

5. Desarrollo Caso de Estudio

5.1 Fundamento Teórico

Se realizará un breve compendio sobre la tecnología SDN, para su comprensión, análisis y el correcto desarrollo de este proyecto. En primer lugar se expondrá los conceptos, arquitectura, ventajas, etc. de las Redes Definidas por Software; luego se analizará el controlador SDN a ser utilizado para finalmente revisar la herramienta de emulación MININET la cual será de fundamental ayuda para demostrar y analizar el funcionamiento de esta nueva tecnología.

5.1.1 Redes Definidas por Software.

En los últimos años, las arquitecturas de red tradicionales han disminuido notablemente su capacidad de brindar soluciones adecuadas o eficientes a problemas tanto de programación como de administración de red y por ende a recuperación de fallos ante eventuales problemas, la tecnología SDN ha empezado a desarrollarse, con el fin de reemplazar dichas arquitecturas y brindar una solución que sea más dinámica, manejable, rentable y por supuesto adaptable a cualquier tipo de red, inclusive si se trata de manejar soluciones que incluyan equipos de diferentes fabricantes.

Con una red SDN, el administrador puede darle forma al tráfico desde una consola de control centralizado sin tener que ir configurando cada uno de los diferentes equipos de networking de los que se disponga. El administrador puede cambiar el comportamiento de los dispositivos de la red en cualquier momento asignando o quitando prioridad, o inclusive bloqueando tipos específicos de paquetes con un nivel de control detallado.

5.1.1.1 Historia.

En los inicios de internet, las alternativas para solucionar el problema del crecimiento exponencial de internet resultaron ser procesos muy lentos, tales como la programación de nuevos protocolos de red, ya que debían ser estandarizados. Fue por este motivo que los investigadores apostaron por desvincular las funciones de control y reenvío de la red, permitiendo la programación de la misma y quedando abstraída de la infraestructura de las aplicaciones y los servicios.

Más adelante, el aumento del volumen de tráfico planteó nuevos retos a los desarrolladores que buscaron soluciones capaces de gestionar el tráfico de manera eficiente, es así que en el 2007 el proyecto Ethane presenta una arquitectura centraliza cuyo control se realiza a través de políticas que permiten mayor granularidad y gestión de la red.

Estos precedentes históricos se convirtieron en las bases de un nuevo protocolo llamado OpenFlow, una pieza clave para la construcción de soluciones SDN ya que permite que éstas sean directamente ágiles, programables y centralizadas.

5.1.1.2 Definición.

Las Redes Definidas por Software o SDN (Software Defined Network) son una arquitectura dinámica, manejable, rentable y adaptable, la cual quita la potestad del manejo del flujo de paquetes a los dispositivos de red y se le da a una aplicación de software llamada controlador, en otras palabras, desacopla el plano de datos del plano de control de los dispositivos y el controlador es el responsable de gestionar la información de reenvío de paquetes dentro de una red.

En una red tradicional el tráfico es gestionado de acuerdo a las reglas especificadas en el firmware del dispositivo, razón por la cual todos los paquetes son tratados de la misma forma, sin poder dar un trato prioritario o diferente según los requerimientos de red que se presenten en ese momento, haciendo una analogía, cuando en la vida real un camión sigue una ruta predefinida hacia un destino, no está exento de problemas de congestión o cierre de la vía, por tanto la solución sería dar un aviso oportuno y proponer una ruta alterna en el momento mismo del incidente, de esta forma se obtiene una mayor eficiencia en cuanto a tiempos de respuesta, retrasos, uso de recursos, etc. SDN se plantea para solucionar este inconveniente, enviando los datos en forma de flujos al controlador, el cual tiene la capacidad de programar el comportamiento de la red a través de software de una manera ágil, ya que un administrador de red puede darle forma al tráfico desde una consola de control centralizada sin tener que tocar dispositivos de red de manera individual.

5.1.1.3 Arquitectura.

Como ya se mencionó anteriormente las Redes Definidas por Software se caracterizan por la división entre el plano de datos y el de control, dada esta división SDN establece una arquitectura de tres capas: Capa de aplicación, Capa de Control y Capa de Infraestructura, tal como se muestra en la figura a continuación.

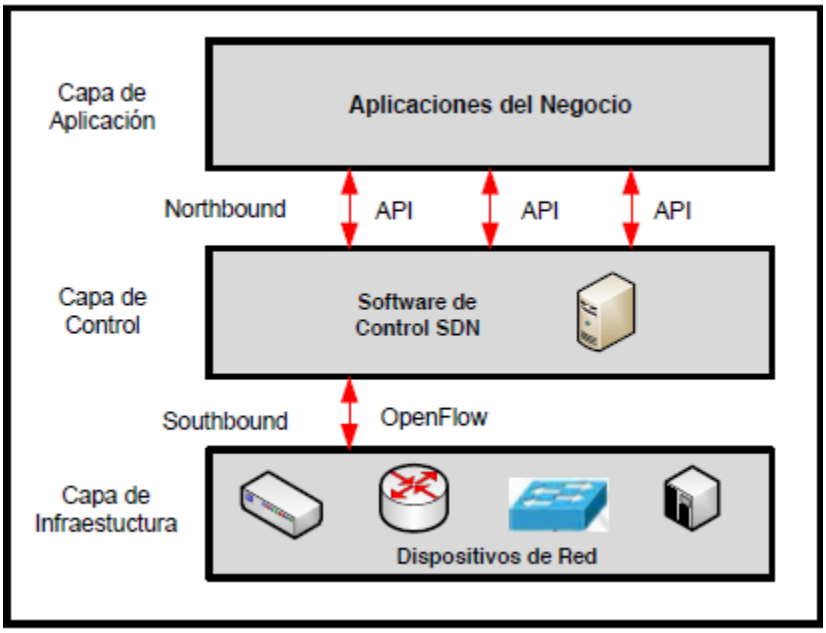

Figura 1. Arquitectura SDN

La *Capa de Aplicación*, también conocida como capa de negocio, tiene relación directa con las aplicaciones o servicios de alto nivel requeridos por los usuarios (QoS, firewall, etc). Las instrucciones de las aplicaciones son abstraídas y enviadas al controlador mediante el uso de lo que se conoce como interfaz de northbound.

La *Capa de Control*, es la encargada de establecer el comportamiento de los flujos de red; el controlador permite la transmisión de las instrucciones hacia los dispositivos a través de lo que se conoce como interfaces de southbound, la más conocida OpenFlow. Esta capa se encuentra ligada con el plano de control.

La *Capa de Infraestructura*, está conformada por todos los dispositivos físicos de comunicación (routers, switches, etc.), los mismos que administran las tablas de flujo en función de las indicaciones del controlador. Esta capa se encuentra relacionada con el plano de datos.

5.1.1.4 Ventajas.

La implementación de una SDN tiene varios beneficios en comparación a los esquemas de red actuales, los mismos que se detallan a continuación:

- Gestión centralizada de la red, todas las configuraciones para el tratamiento del tráfico que fluye por la red se la realiza en el controlador SDN.

- Se basa en estándares abiertos, con lo cual el administrador de la red es capaz de manipular la misma e implementar mejoras por sí mismo, sin la necesidad de esperar a que los fabricantes de sus equipos introduzcan nuevas tecnologías, obteniendo de esta manera una independencia de los vendedores de equipos de conectividad.

- La automatización y la gestión de la red a través de Interfaces de Programación de Aplicaciones conocidas comúnmente como API’s son mejoradas.

- Capacidad para administrar dinámicamente recursos de red bajo demanda (como reserva de ancho de banda) según necesidades de cada recurso y tipo de usuario, con lo que las redes están al servicio del negocio y no al revés.

- Posibilidad de gestionar de forma centralizada todos los recursos de seguridad de la empresa.

- Mayor agilidad en el despliegue y control de redes, lo que permite afrontar rápidamente cualquier cambio en el negocio y construir entornos de innovación de forma ágil.

- Se adaptan sin ningún esfuerzo y en menor tiempo a cambios o requisitos que un cliente necesita, brindando de esta manera un mejor servicio al usuario final.

5.1.2 Controladores SDN.

El controlador SDN al manejar toda la inteligencia de la red, se convierte en la parte central de la arquitectura SDN, por tanto es en donde el administrador se encarga de definir de manera sencilla y rápida las políticas o reglas que administrarán el flujo de datos; convirtiéndose en una ventaja frente a las redes tradicionales.

5.1.2.1 Controladores de código abierto.

En la siguiente tabla comparativa se muestran algunos de los controladores de código abierto con las características más relevantes de cada uno de ellos.

	NOX	POX	FLOODLIGHT	OPENDAYLIGHT	BEACON	TREMA	RYU
LENGUAJE DE DESARROLLO	C ++	Python	Java	Java	Java	Rudy/C	Python
RENDIMIENTO	Rápido	Lento	Rápido	Rápido	Rápido	Rápido	Lento
SOPORTE OPENFLOW	OF v1.0	OF v1.0	OF v1.0	OF v1.0, v1.3	OF v1.0	OF v1.3	OF v1.0, v1.2, v1.3
VIRTUALIZACION	Mininet y Open vSwitch	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
SOPORTE DE PLATAFORMAS	Linux	Linux, Mac OS, Windows	Linux, Mac OS, Windows	Linux, Mac OS, Windows	Linux, Mac OS, Windows y Android para móviles	Linux	Linux
CURVA DE APRENDIZAJE	Moderada	Fácil	Compleja	Compleja	Moderada	Compleja	Moderada
DISTRIBUIDO	No	No	Si	Si	Si	No	Si

Figura 2. Características controladores SDN

De los controladores SDN mostrados en el cuadro anterior se escoge el controlador POX, ya presenta una curva de aprendizaje rápida, tiene buena documentación y permite desarrollar los módulos de una manera muy ágil.

5.1.2.1.1 Controlador POX.

POX es un controlador OpenFlow, considerado como una nueva versión de NOX basado en Python, desarrollado para cubrir los requerimientos de las SDN siendo a la vez uno de los frameworks de desarrollo constante y creciente, que permite escribir un controlador OpenFlow.

POX tiene una API SDN de alto nivel que incluye un gráfico de topología con capacidad de consulta y soporte para la virtualización

Algunas de las características más importantes de POX son:

- Interfaz OpenFlow basada en Python.
- Está dirigido específicamente a Linux, Mac OS y Windows.
- Interfaz gráfica y herramientas de visualización similares a las de NOX.
- Incluye soporte especial para Open VSwitch.

5.1.2.2 Controladores comerciales.

Si bien los controladores SDN de código abierto fueron de los primeros en salir, varios proveedores han comenzado a ofrecer controladores SDN comerciales como parte de sus portafolios de redes programables.

A continuación se va a detallar algunos de los controladores SDN que las principales compañías han sacado al mercado:

- **APIC (Application Policy Infrastructure Controller)**, es un controlador lanzado al mercado por Cisco que ofrece control centralizado al nivel de aplicaciones para la automatización, seguridad y los servicios de red. El objetivo principal de APIC es proveer de autoridad política y de mecanismos de resolución de políticas a los dispositivos Cisco ACI para optimizar el desempeño de las aplicaciones y la eficiencia de la red.

- **VAN (Virtual Application Networks)**, es el controlador SDN de HP diseñado para funcionar en campus, datacenter o proveedores de servicios; este controlador proporciona un punto de control unificado en una red habilitada para OpenFlow, simplificando la administración, el aprovisionamiento y la orquestación, permitiendo la entrega de una nueva generación de servicios de red basados en aplicaciones. El controlador VAN también puede agruparse, lo que permite a un controlador hacerse cargo de las funciones de otro si uno falla.

- **ProgrammableFlow PF6800**, este controlado desarrollado por NEC, proporciona una tecnología de última generación que transforma la red de hoy, con lo que los beneficios de la virtualización se convierten en conmutación de alto rendimiento y se habilitan las redes orientadas a las aplicaciones. El PF6800 utiliza la tecnología OpenFlow para

separar el plano de control del plano de datos de la red, lo que permite la virtualización de toda la red.

- **VSC (Virtualized Services Controller)**, este producto de Nuage Networks, virtualiza la infraestructura de red del centro de datos y conecta recursos de cómputo cuando se crean, es una solución abierta y escalable que busca satisfacer las demandas de los centros de datos.

- **NSX,** es el controlador utilizado por VMware que ofrece un modelo operativo totalmente nuevo para la red que conforma la base del centro de datos definido por software. Dado que NSX crea redes en el software, los operadores del centro de datos pueden lograr unos niveles de agilidad, seguridad y economía que antes eran imposibles con las redes físicas

5.1.3 Protocolo Open Flow.

El protocolo OpenFlow fue originalmente propuesto como una alternativa para el desarrollo de protocolos experimentales en el campus de la universidad de Standford, en el 2008 Martin Casado, un estudiante de dicha Universidad en Silicon Valley, California, desarrolló "Ethane", una arquitectura de red lógicamente centralizada para la gestión de las políticas de seguridad de las redes empresariales; esta idea condujo a lo que actualmente se conoce como OpenFlow.

OpenFlow se define como uno de los primeros estándares y la primera interfaz de comunicaciones estándar entre los planos de control y datos de una arquitectura de SDN. Este protocolo permite el acceso directo y manipulación del plano de reenvío de los dispositivos de red tales como switches y routers, tanto físicos como virtuales.

La OFN (Open Networking Foundation), es la encargada de la estandarización de OpenFlow y lo hace a través de grupos de trabajo técnicos encargados de la configuración, pruebas de interoperabilidad, y otras actividades del protocolo, ayudando a garantizar la interoperabilidad entre los dispositivos de red y software de control de diferentes proveedores.

5.1.3.1 Funcionamiento.

La secuencia de paquetes que atraviesan una red compartiendo campos de su cabecera de datos se denomina flujo, OpenFlow introduce este concepto de flujos y traslada la decisión de reenvío de paquetes de los switches a los controladores. Existe una aplicación que se ejecuta en la interfaz del controlador que está unida a todos los switches de la red con el fin de facilitar la configuración y administración del flujo de datos.

Cuando el primer paquete de un flujo llega al switch, este verifica las tablas de flujo con el objetivo de encontrar coincidencias, si ninguna coincidencia es encontrada se debe crear un nuevo flujo, esta responsabilidad la tiene el controlador el cual define el nuevo flujo para ese paquete y crea una o más entradas en la tabla del switch; luego de insertar esta entrada los

paquetes que coincidan con la nueva entrada añadida se envían directamente a su destino sin la necesidad de enviarlo al controlador

OpenFlow ha identificado las funciones comunes de la estructura de un switch Ethernet tradicional de los diferentes fabricantes, con la finalidad de programar tablas de flujo que puedan acoplarse a cualquier switch de cualquier marca.

5.1.3.2 Switch OpenFlow.

Un switch OpenFlow está compuesto de un canal seguro para comunicarse con el controlador, de una o más tablas de flujo y una tabla de grupo, tal como se muestra en la figura a continuación.

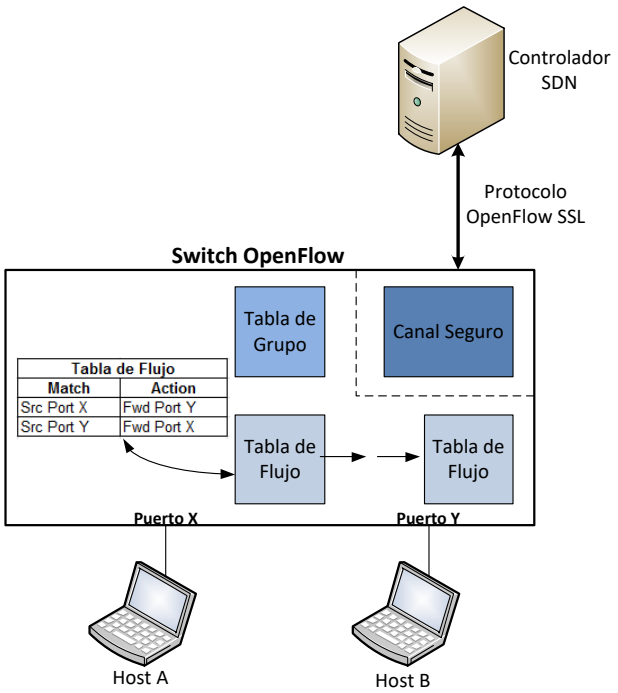

Figura 3. Elementos Switch OpenFlow

- **Tabla de Flujo**: Utilizando el protocolo OpenFlow, el controlador puede agregar, actualizar y eliminar entradas de flujo en las tablas, cada tabla de flujo consta de campos para emparejamiento, contadores de actividad y un set de instrucciones para aplicar a los paquetes coincidentes.

- **Tabla de Grupo**: Una tabla de grupo está compuesta por entradas de grupo.

- **Canal Seguro:** Es la interfaz encargada de conectar el switch OpenFlow con el controlador, el servidor controlador administra, recibe eventos y envía paquetes al switch a través de esta interfaz.

5.1.3.3 Versiones.

Desde que fue desarrollado el protocolo se han realizado cambios y mejoras sobre el mismo, a continuación las versiones que se han ido desarrollando:

- OpenFlow 1.0 (diciembre 2009: El controlador se comunica con el switch mediante el protocolo OpenFlow utilizando un canal seguro TLS11 y una tabla de flujo.

- OpenFlow 1.1 (febrero 2011): introduce el soporte multitabla y la forma para vincular los resultados de las tablas individuales, se añade soporte para vlan's, MPLS y puertos virtuales.

- OpenFlow 1.2 (diciembre 2011): se añade el soporte para reescritura de paquetes y soporte para IPV6.

- OpenFlow 1.3 (junio 2012): se puede manejar la cabecera extendida de IPV6 y se puede filtrar eventos de cada conexión para optimización dentro de un ambiente con varios controladores y switches.

- OpenFlow 1.4 (octubre 2013): El protocolo se vuelve extensible mediante el uso de TLV15 y mejora la interacción entre el controlador y el switch con la adición de un monitoreo por cada flujo

5.1.4 Emulador de Redes Mininet.

El uso de entornos de simulación es algo imprescindible para el desarrollo de cualquier proyecto en ingeniería mínimamente complejo. Mediante simulación es posible reproducir el comportamiento de lo que se ha implementado en unas condiciones muy parecidas a la realidad, permitiendo depurar y detectar errores de alto nivel antes de proceder al despliegue. En términos académicos y de investigación, aún es más evidente su importancia, donde los recursos son limitados y los costes deben reducirse al máximo. En particular, el presente proyecto se mueve

dentro del ámbito de las telecomunicaciones, con hardware dedicado realmente caro, lo que supone un problema enorme a la hora de reproducir escenarios, tanto por los costes como por la infraestructura necesaria, lo que convierte a los simuladores de red en un elemento esencial e imprescindible.

5.1.4.1 CONCEPTO.

Mininet no es un simple emulador, es más bien un sistema de orquestación de emulación de una red, que permite la rápida creación de escenarios virtuales en una sola computadora. Utiliza una virtualización ligera basada en Linux para hacer que un único sistema se vea como una red completa, ejecutando el mismo kernel, sistema y código de usuario. Los hosts, switches, enlaces y controladores virtuales de Mininet se crean con software y no con hardware, y en su mayor parte su comportamiento es similar a elementos de hardware discretos.

Este emulador, se ha convertido en la herramienta preferida para emular Redes Definidas por Software debido a su flexibilidad, sus facilidades de despliegue, administración e intercambio de escenarios, interactividad, escalabilidad y ancho de banda para pruebas en el orden de los 2 Gbps, etc.

5.1.4.2 Características.

- Rapidez: la generación de una red sencilla tarda unos pocos segundos.

- Flexibilidad: topologías y características nuevas se pueden setear por software usando lenguajes de programación y SO comunes.

- Realista: se puede ejecutar programas como WireShark, entre otros, para análisis de trafico ya que representando un comportamiento real con alto grado de confianza.

- Amigable: es fácil de usar, se puede crear y ejecutar experimentos en Mininet escribiendo scripts sencillos.

- Escalabilidad: Es escalable a redes grandes con un solo computador.

- Aplicabilidad: Correctas implementaciones (prototipos) se pueden implementar en redes basadas en Hardware sin cambiar su código fuente.

- Compartible: Prototipos pueden ser compartidos para que otros colaboradores y desarrolladores puedan correr y modificar el experimento.

- Interactividad: Administración y simulación ocurren en tiempo real.

5.2 Diseño y desarrollo.

En la sección anterior se realizó una exposición resumida del marco teórico referente a las Redes Definidas por Software, con el fin de obtener los elementos necesarios para realizar la

simulación de una SDN utilizando la herramienta Mininet. A continuación se detallarán las herramientas y configuraciones necesarias para llevar a cabo dicha simulación.

5.2.1 Componentes.

Se realizará una descripción de los elementos físicos y software utilizados en este proyecto.

5.2.1.1 Elementos de hardware.

El programa emulador Mininet, se lo puede instalar de dos maneras:

- En una herramienta de virtualización, importando la máquina virtual (archivo ovf) que se obtiene en el sitio web oficial de Mininet, ó

- En una PC con sistema operativo Ubuntu, en la cual se debe descargar todos los paquetes del programa Mininet de su repositorio.

Para cualquiera de los dos casos de instalación explicados anteriormente, no existe por parte del propietario del emulador requisitos mínimos de hardware para su instalación y buen

funcionamiento. En este proyecto se ha decidido utilizar la máquina virtual, debido a flexibilidad y agilidad que presenta esta opción para alcanzar los objetivos propuestos.

La PC portátil donde se va instalar el emulador Mininet tiene las siguientes características:

- Marca y Modelo: HP Pavillion 14 Notebook.
- Procesador: Intel Core (TM) i7 2.6 Ghz.
- Memoria RAM: 8 GB.
- Sistema Operativo: Windows 10.

5.2.1.2 Elementos de software.

Para la puesta en marcha de la red SDN, se apoyará de los siguientes programas: VM Mininet. VMware Workstation y Securete CRT.

5.2.1.2.1 Mininet.

El primer paso para la instalación es descargar la imagen de la máquina virtual, la misma que tiene el entorno ya configurado y listo para que todo funcione adecuadamente, este software se podrá en: http://mininet.org/download/. El archivo descargado viene en formato zip, y contiene dos archivos, uno con extensión *.ovf que es el que contiene todos los parámetros de configuración referentes a la máquina virtual, y otro con extensión *.vmdk, que es el disco de datos virtual.

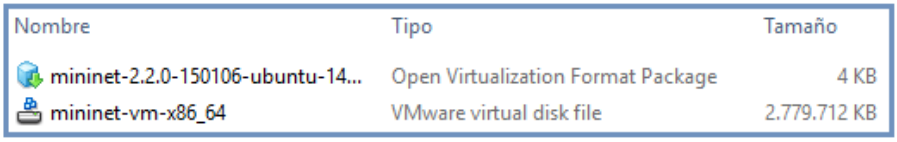

Figura 4. Archivos de instalación VM Mininet

5.2.1.2.2 Vmware workstation.

El siguiente paso es ejecutar la máquina virtual descargada, mediante un software de virtualización, para este caso el programa a utilizar es VMware Workstation el mismo que se podrá de http://www.vmware.com/products/workstation.html. A continuación se describe los pasos a seguir para una correcta configuración del software de virtualización y ejecución de la máquina virtual Mininet:

1) Ejecutar el archivo *.exe descargado y seguir los pasos de instalación.

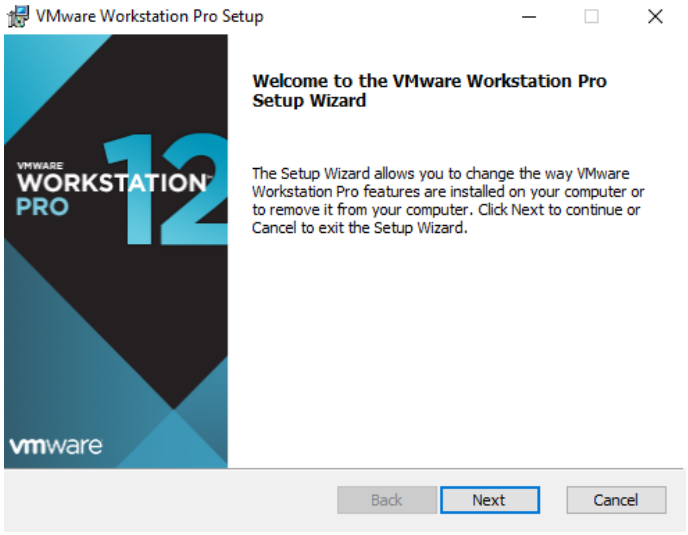

Figura 5. Pantalla de inicio de instalación VMWare Workstation

2) Una vez instalado VMware Workstation, se debe importar la máquina virtual seleccionando la opción “Open a Virtual Machine” y posteriormente escogiendo el archivo *.ovf de donde se lo tenga descargado.

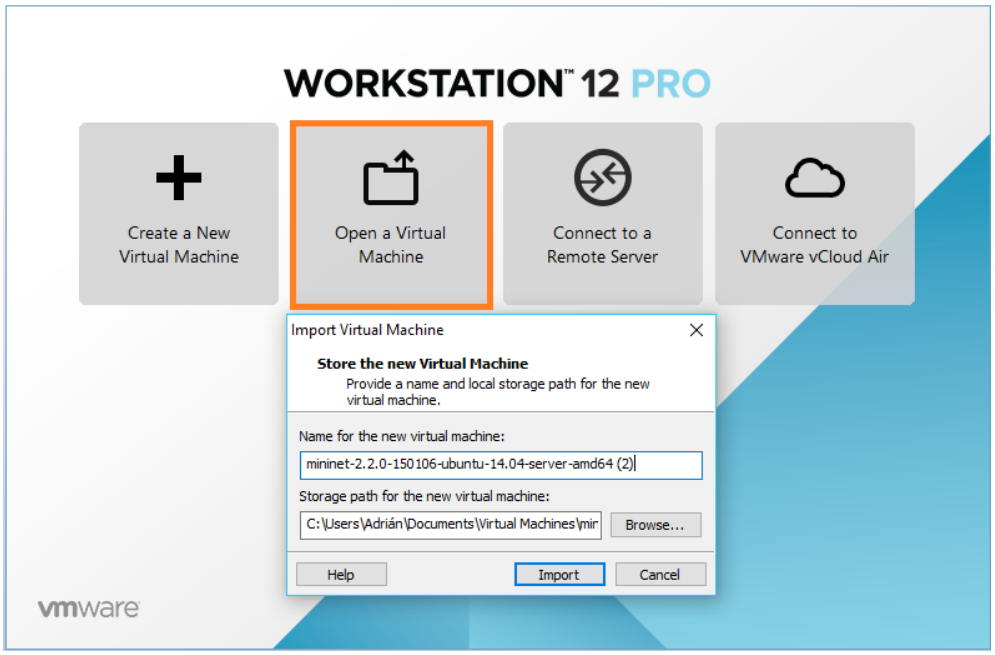

Figura 6. Importación de VM Mininet

3) Luego de importar la máquina virtual, es necesario habilitar otra tarjeta de red en modo “Host Only”, para poder ingresar mediante SSH a dicha máquina por la IP de dicha tarjeta.

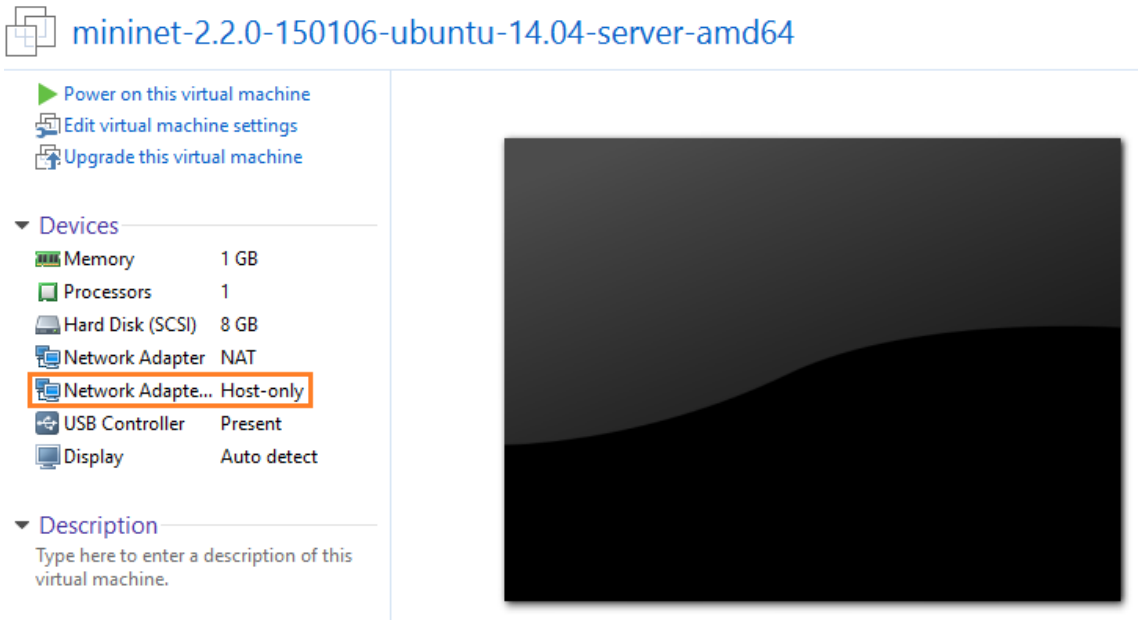

Figura 7. Integración nueva tarjeta de red

4) Finalmente se enciende la máquina virtual dando click en “Power on this virtual machine”, y se ingresa a la misma con el user/pass: mininet.

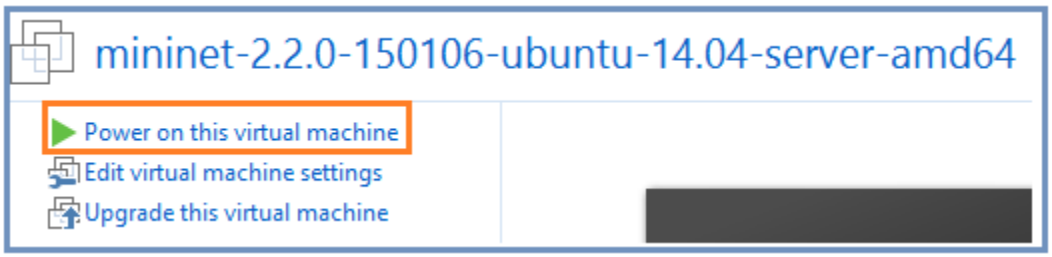

Figura 8. Encendido de VM Mininet

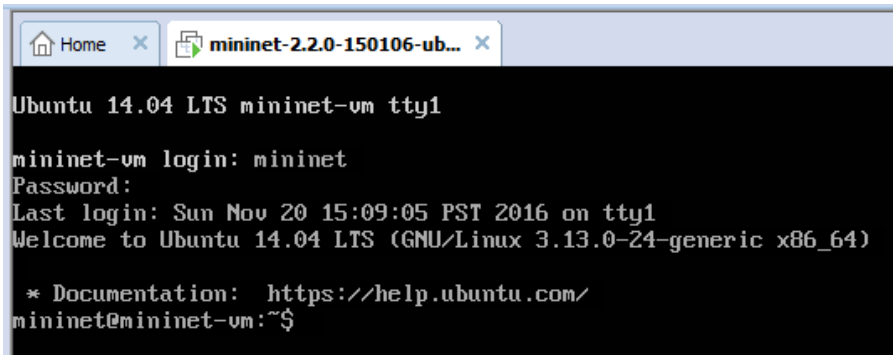

Figura 9. Ingreso a VM Mininet

Algo importante que se debe realizar para el ingreso mediante SSH a la máquina virtual es habilitar la tarjeta "Host Only" que se agregó en los pasos anteriores, para lo cual se ejecuta el siguiente comando:

mininet@mininet-vm:~$ **sudo dhclient eth1**

Una vez ejecutado se comprueba si la tarjeta está habilitada, con el comando:

mininet@mininet-vm:~$ **ifconfig**

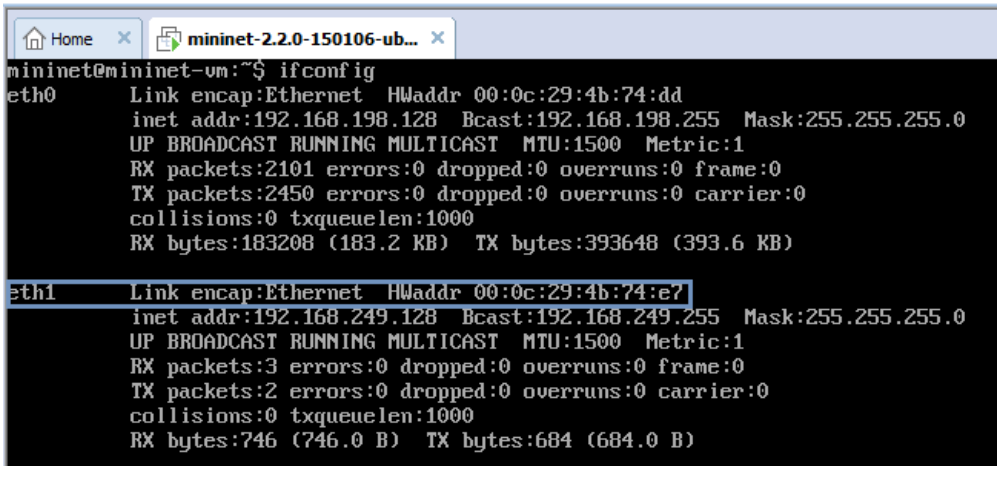

Figura 10. Tarjetas de red habilitadas

5.2.1.2.3 SECURETE CRT

Para el ingreso vía SSH a la máquina virtual se ocupará el programa SecureteCRT, en el cual una vez instalado se debe configurar una sesión para el ingreso a dicha máquina; configurada la sesión, se dá click en Connect, se ingresa el password y se obtiene la conexión a la máquina virtual a través de SSH.

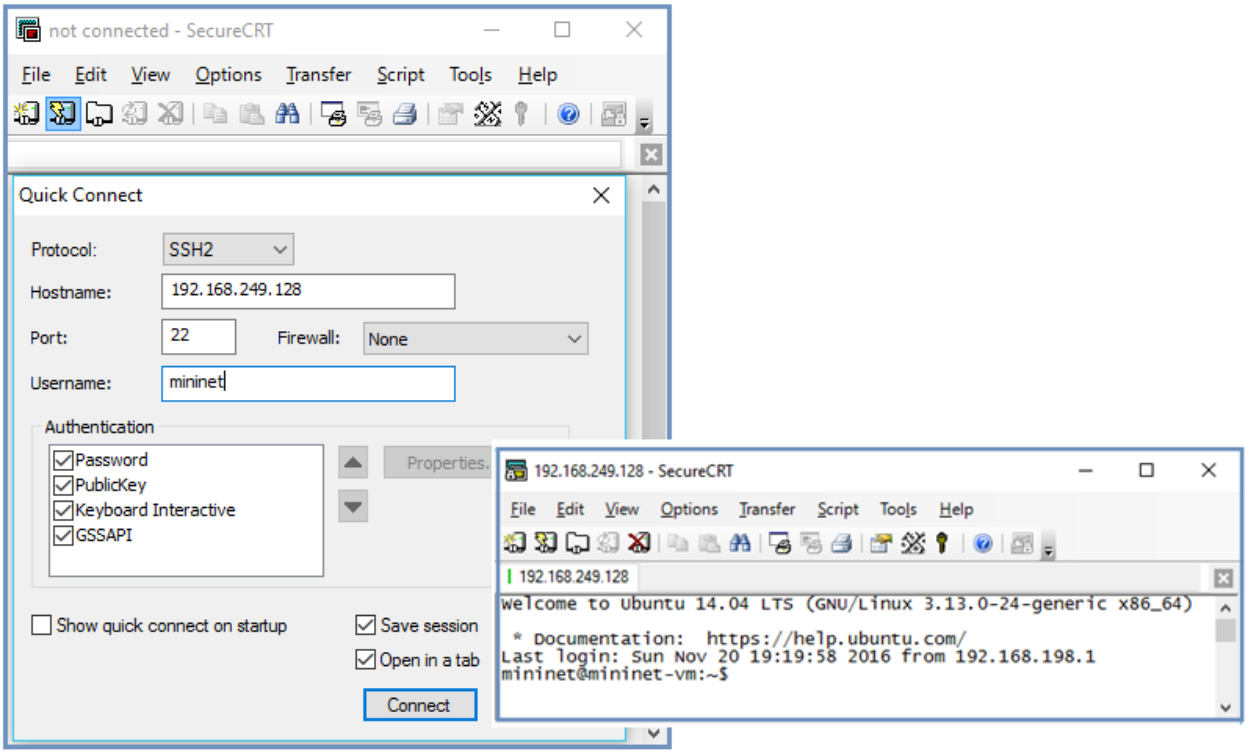

Figura 11. Configuración de sesión e ingreso via SSH mediante SecureCRT

5.2.2 Topología usada para la simulación.

La topología a utilizar, sobre la cual se demostrará la funcionalidad del Firewall L2 SDN es la siguiente:

NOMBRE	IP	MAC
HTTP	10.0.0.1	00:00:00:00:00:01
MAIL	10.0.0.2	00:00:00:00:00:02
MNET	10.0.0.3	00:00:00:00:00:03
Host1	10.0.0.4	00:00:00:00:00:04
Host2	10.0.0.5	00:00:00:00:00:05
Host3	10.0.0.6	00:00:00:00:00:06
Host4	10.0.0.7	00:00:00:00:00:07

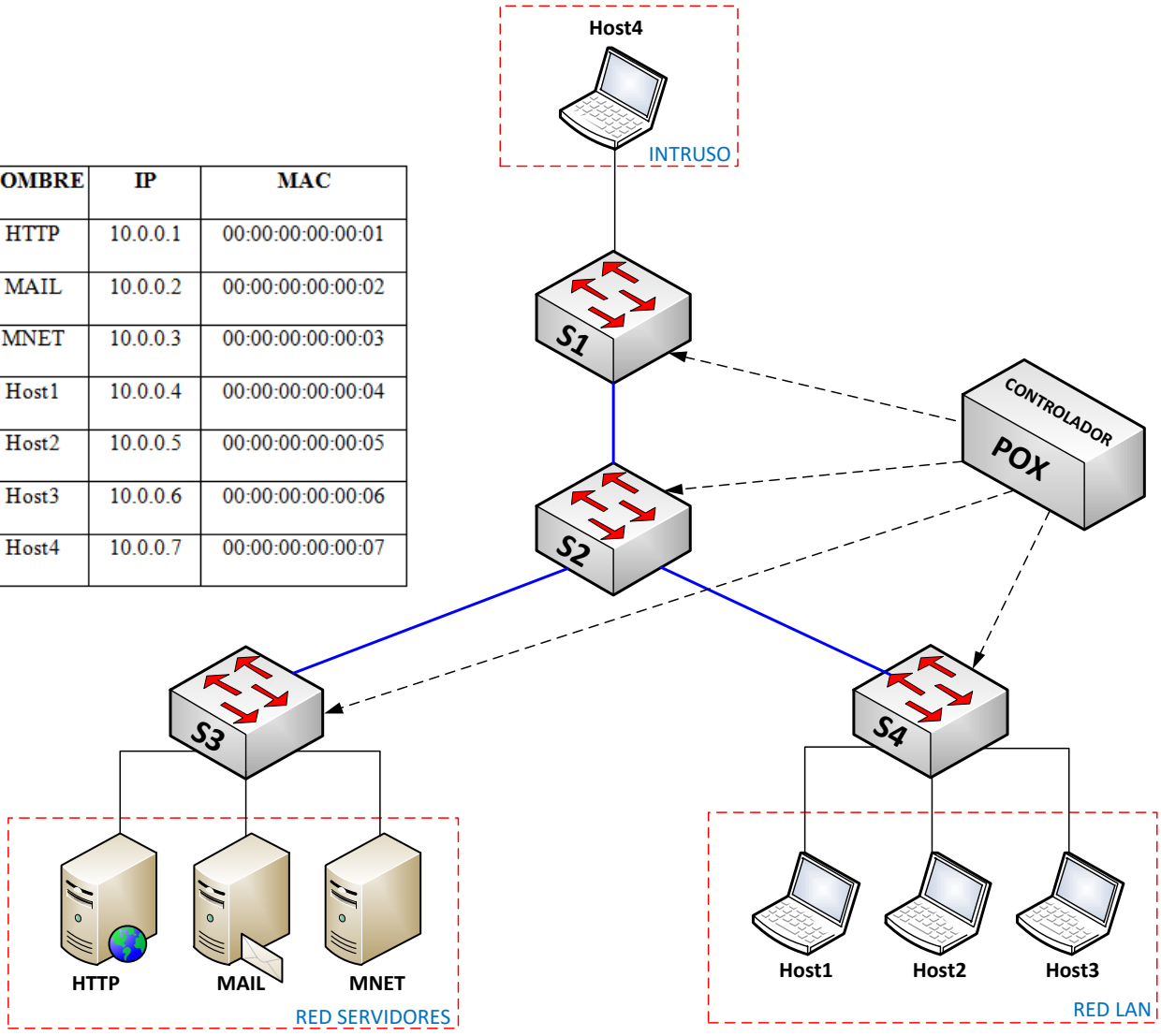

Figura 12. Topología de Red

5.2.2.1 Creación de topología personalizada.

Mininet permite la creación de topologías personalizadas para aquellas situaciones donde se requiere crear escenarios complejos, con enlaces distintos, switches controlados por más de un controlador o asignar propiedades concretas a ciertos dispositivos.

En este proyecto, se apoyará en el archivo topo-2sw-2host.py el cual viene cargado por defecto en la máquina virtual de Mininet dentro del directorio ~/mininet/custom; para desarrollar la topología mostrada en el punto anterior.

El código mostrado a continuación presenta el script para la generación de la topología requerida, el cual ha sido guardado en un archivo con nombre topologia-sdn.py.

```
from mininet.topo import Topo

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

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

        # Initialize topology
        Topo.__init__( self )

        # Add hosts and switches
        Host1 = self.addHost( 'host1' )
        Host2 = self.addHost( 'host2' )
        Host3 = self.addHost( 'host3' )
        Host4 = self.addHost( 'host4' )
        Server1 = self.addHost( 'HTTP' )
        Server2 = self.addHost( 'MAIL' )
        Server3 = self.addHost( 'MNET' )
        Switch1 = self.addSwitch( 'switch1' )
        Switch2 = self.addSwitch( 'switch2' )
        Switch3 = self.addSwitch( 'switch3' )
        Switch4 = self.addSwitch( 'switch4' )

        # Add links
        self.addLink( Host1, Switch4 )
        self.addLink( Host2, Switch4 )
        self.addLink( Host3, Switch4 )
        self.addLink( Host4, Switch1 )
        self.addLink( Switch1, Switch2 )
        self.addLink( Switch2, Switch3 )
        self.addLink( Switch2, Switch4 )
        self.addLink( Switch3, Server1 )
        self.addLink( Switch3, Server2 )
        self.addLink( Switch3, Server3 )


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

Figura 13. Código para creación de la topología

Para la ejecución del script se ejecuta el comando mostrado a continuación, obteniendo el resultado mostrado en la figura 14.

sudo mn --custom topologia-sdn.py --topo mytopo --mac --controller=remote

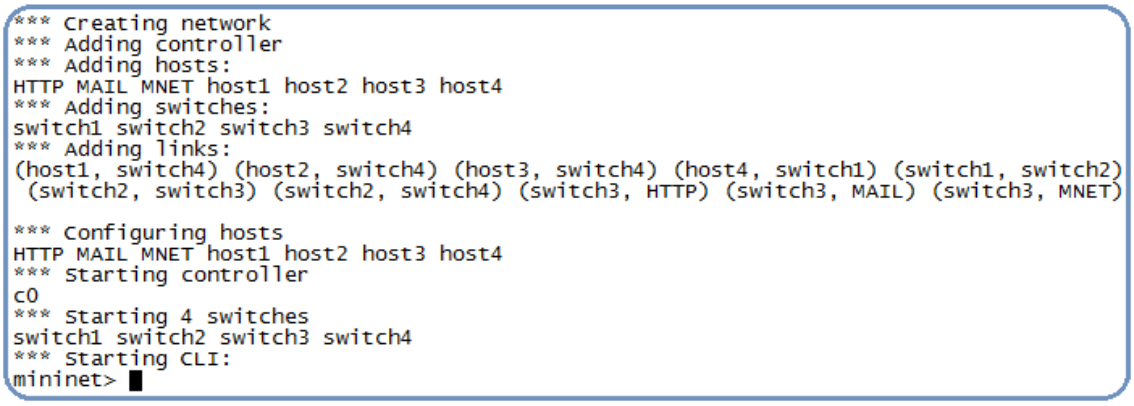

Figura 14. Creación de la topología

5.3 Controlador POX.

Como se ha comentado, POX es una plataforma que permite crear módulos que luego contralarán los diferentes switches que se le asocien. Una vez iniciados estos módulos, cada uno de ellos restará escuchando a una IP y puerto definidos, e interactuará con los diferentes protocoles que vaya escuchando.

Para iniciar uno de los módulos debe hacerse mediante la ejecución del comando **pox.py** que se encuentra dentro la carpeta principal del controlador, por defecto en **~/pox**, pasándole como argumentos todos los módulos que se desean iniciar. POX por defecto trae algunos módulos interesantes ya implementados con algunos comportamientos básicos, como el de un "Hub" o un "Switch L2".

5.3.1 Creación del módulo firewall.

En esta sección se implementará el módulo del Firewall SDN, el mismo que realizará una función similar a la de un ACL (Lista de Control de Acceso), basándose en una lista de direcciones de capa 2 (MAC), que el firewall debe tomar como referencia para bloquear la conexión entre los diferentes hosts.

Al momento de establecerse la conexión entre el controlador y el conmutador, la aplicación de firewall instala las entradas de regla de flujo en la tabla OpenFlow para de ésta manera desactivar toda comunicación entre cada par de direcciones de capa 2 definidas en un archivo *.csv.

Con lo antes mencionado la meta es:

- Bloquear todas las conexiones entre hosts de la granja de servidores (HTTP, MAIL, MNET).

- Permitir que el Host1 y Host2 solo alcancen a los servidores HTTP y MAIL.

- Bloquear todas las conexiones del Host4.

- Permitir que el Host 3 alcance a todos los Host y Servidores excepto el Host4.

A continuación se muestra el código del módulo de Firewall así como las políticas del mismo.

```
from pox.core import core
import pox.openflow.libopenflow_01 as of
from pox.lib.revent import *
from pox.lib.util import dpidToStr
from pox.lib.addresses import EthAddr
from collections import namedtuple
import os

log = core.getLogger()
policyFile = "%s/pox/pox/misc/politicas-firewall.csv" % os.environ['HOME']


class Firewall (EventMixin):

    def __init__(self):
        self.listenTo(core.openflow)
        self.rules = []
        with open(policyFile, 'r') as f:
            for line in f:
                try:
                    rule = line.strip().split(',')
                    if rule[0] != 'id':
                        self.rules.append((EthAddr(rule[1]), EthAddr(rule[2])))
                except:
                    pass
        log.debug("Enabling Firewall Module")

    def _handle_ConnectionUp(self, event):
        for src, dst in self.rules:
            msg = of.ofp_flow_mod()
            msg.match = of.ofp_match()
            msg.match.dl_src = src
            msg.match.dl_dst = dst
            msg.priority = 65535
            event.connection.send(msg)
        log.debug("Firewall rules installed on %s", dpidToStr(event.dpid))


def launch():
    '''
    Starting the Firewall module
    '''
core.registerNew(Firewall)
```

Figura 15. Código del archivo firewall_l2.py

”

```
id,mac_0,mac_1
1,00:00:00:00:00:07,00:00:00:00:00:01
2,00:00:00:00:00:07,00:00:00:00:00:02
3,00:00:00:00:00:07,00:00:00:00:00:03
4,00:00:00:00:00:07,00:00:00:00:00:04
5,00:00:00:00:00:07,00:00:00:00:00:05
6,00:00:00:00:00:07,00:00:00:00:00:06
7,00:00:00:00:00:03,00:00:00:00:00:04
8,00:00:00:00:00:03,00:00:00:00:00:05
9,00:00:00:00:00:07,00:00:00:00:00:02
10,00:00:00:00:00:07,00:00:00:00:00:02
11,00:00:00:00:00:01,00:00:00:00:00:02
12,00:00:00:00:00:02,00:00:00:00:00:03
13,00:00:00:00:00:01,00:00:00:00:00:03
```

Figura 16. Código del archivo politicas-firewall.csv

5.4 Simulación, pruebas y resultados.

Una vez establecida la red en el simulador MiniNet así como el módulo Firewall, se procede con la simulación y las pruebas respectivas para verificar si se cumple con la meta propuesta en las políticas del firewall.

5.4.1 Ejecución simulación.

Para que el escenario funcione como se espera, será necesario el uso de dos módulos: uno responsable de permitir la comunicación entre los diversos host de la red (l2_learning.py), y otro responsable de aplicar las políticas de Firewall (firewall_sdn.py). El procedimiento que se realiza es el siguiente:

1. Abrir 2 conexiones SSH hacia la máquina virtual de Mininet desde el programa SecureCTR.

2. En la primera conexión se ejecuta el controlador POX y sus módulos aplicando los comandos mostrados a continuación.

```
$ cd ~/pox
$ ./pox.py forwarding.l2_learning misc.firewall_l2
```

3. En la segunda conexión se ejecuta en el CLI de Mininet el siguiente comando para crear la topología.

```
$ sudo mn --custom topologia-sdn.py --topo mytopo --mac --controller=remote
```

5.4.2 Test de comportamiento.

Se realizarán 2 pruebas para comprobar la conectividad entre los diferentes hosts, se espera que un principio todos los host sean accesibles entre ellos y luego, después de aplicar las políticas de firewall, algunos de ellos pierdan conectividad entre sí.

5.4.2.1 Test sin firewall.

Con este test se va a comprobar la conectividad entre los elementos de la red sin aplicar ninguna política de firewall, lo que se traduce a tener el archivo firewall-policies.csv vacío, para lo cual se crea la topología y se ejecuta solo el módulo l2_learning

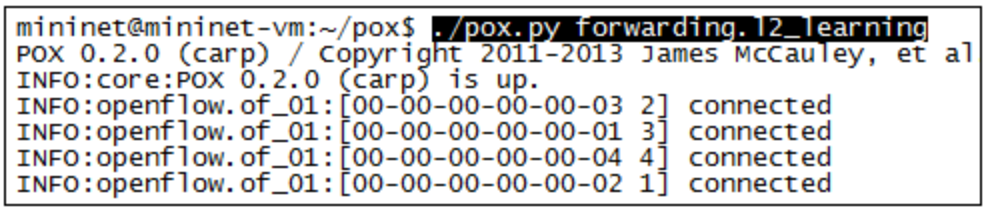

Figura 17. Ejecución controlador POX y modulo l2_learning

A través del comando **ping** ejecutado en la consola de MiniNet se verificará si efectivamente existe conectividad entre los dispositivos, como se detalla en la figura a continuación.

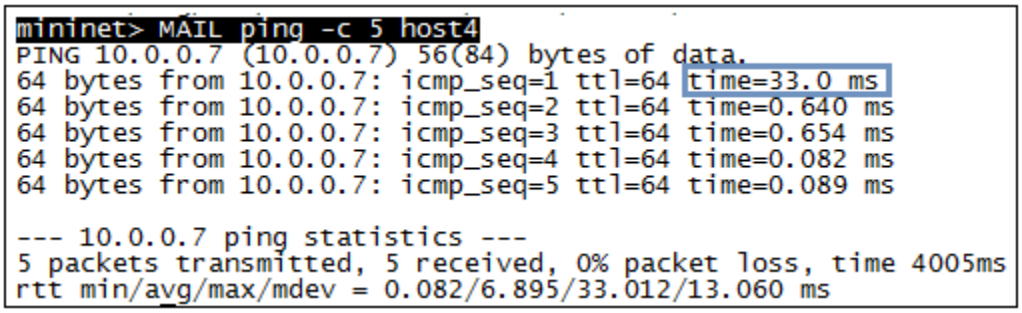

Figura 18. Pruebas de ping entre 2 hosts

Con ayuda del comando **pingall**, se puede comprobar de una manera más rápida la conectividad entre hosts.

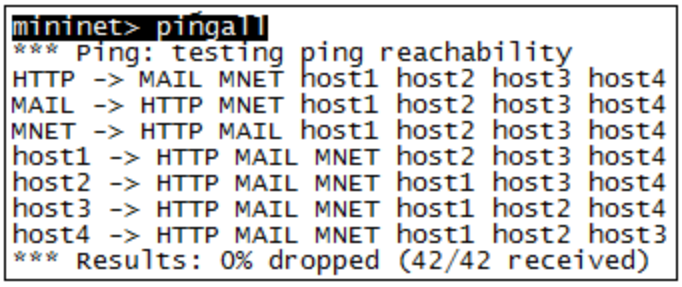

Figura 19. Pruebas de ping entre todos los hosts

Como se puede apreciar en la gráfica anterior hay conectividad entre todos los hosts.

5.4.2.2 Test con firewall.

Después de comprobar conectividad de todos los hosts en el punto anterior, ahora se va a activar la política de firewall para verificar el bloqueo de los hosts que se propuso.

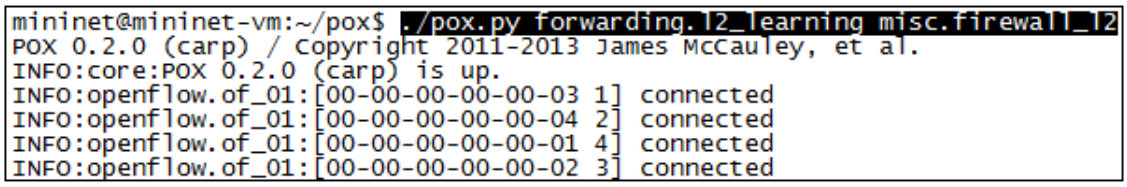

Figura 20. Ejecución controlador POX con los módulos l2_learning y firewall_l2

Nuevamente se utiliza el comando **ping** y luego **pingall**, para verificar la conectividad de los dispositivos y como se puede observar en las siguientes figuras, las políticas del firewall se

aplicaron restringiendo la comunicación de los hosts, esto debido a que todo paquete procedente de un host a otro y que coincida con dichas políticas será descartado por el switch.

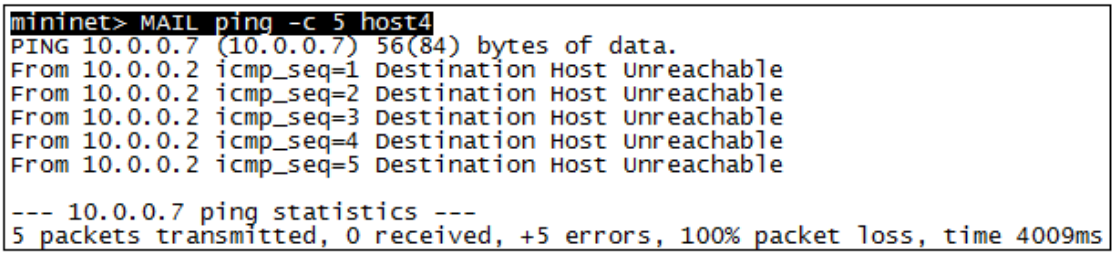

Figura 21. Pruebas de ping entre 2 hosts con firewall aplicado

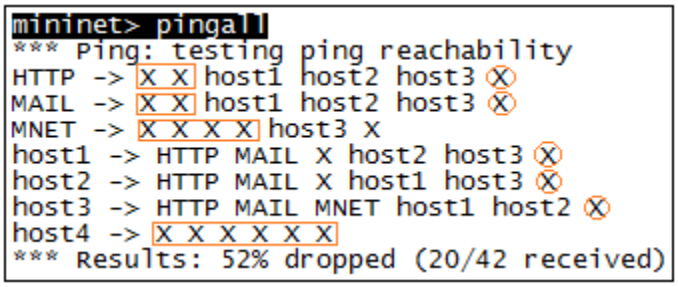

Figura 22. Pruebas de ping entre todos los hosts con firewall aplicado

5.4.3 Análisis de resultados.

5.4.3.1 Análisis test 1.

En lo que respecta al primer test realizado (sin firewall), se puede comprobar que efectivamente hay conectividad entre todos los hosts, pero lo importante a analizar en este punto

es el tiempo de retardo que se observa en el primer paquete enviado, como se puede apreciar en la siguiente figura.

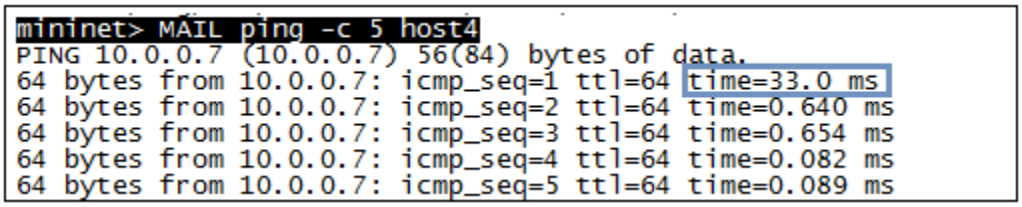

Figura 23. Tiempo de respuesta alto en el primer paquete de ping

Esto es debido a que el primer paquete ICMP que se envía, al no tener los switches de la red ninguna regla que encaje en sus tablas, el paquete viaja hasta el controlador para que éste decida cómo proceder con el mismo, una vez que el controlador recibe el paquete, se guarda del switch que lo ha mandado: su identificador, la interfaz por la que ha llegado el paquete y la MAC de origen del paquete; esta información es almacenada en una tabla que le da conocimiento al controlador sobre en qué interfaces están localizadas cada una de las MAC de la red.

Luego, cuando el controlador reciba un paquete de un destino MAC presente en dicha tabla, el controlador será capaz de instalar las reglas permanentes necesarias en las tablas de flujo de cada switch, para que esa conexión en concreto sea posible con tiempos de delay por dejajo de 1.0 ms.

Con ayuda del comando DPCTL, se puede observar el flujo que se produce en los switches:

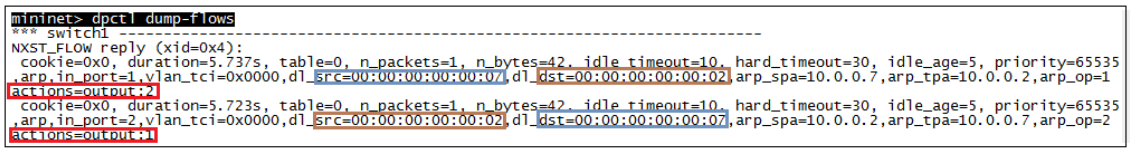

Figura 24. Flujo de paquetes en switch OpenFlow

5.4.3.2 Análisis test 2.

En cuanto al comportamiento del módulo firewall se obtiene los resultados esperados, como se puede apreciar en el test 1, todos los hosts tienen conectividad entre ellos, mientras que en el test 2 los hosts definidos en la política de firewall pierden la capacidad de conectarse como se muestra a continuación:

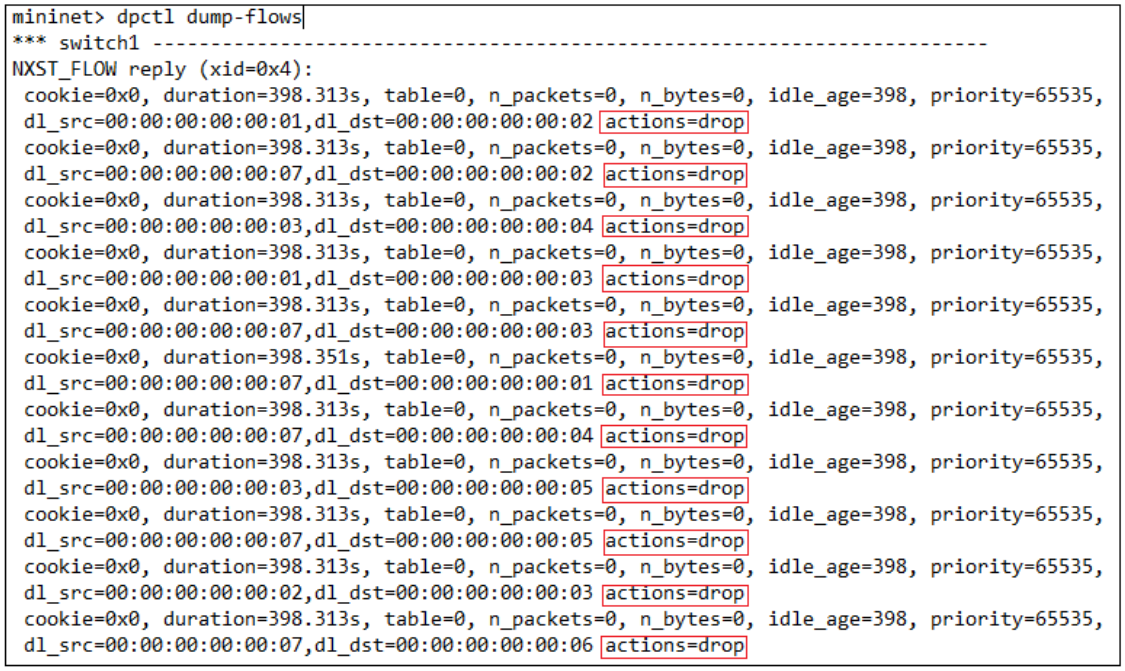

Figura 24. Flujo de paquetes en switch OpenFlow con firewall

6. Conclusiones y Recomendaciones

- Una vez entendida la idea de las Redes Definidas por Software se llegó a implementar un prototipo de un firewall de capa 2, compuesto por un controlador POX con su módulo de firewall, cuatro switches o conmutadores y siete hosts, mediante el cual se pudieron cumplir los objetivos que se plantearon en este proyecto; constatando de esta manera el gran beneficio que brinda una SDN al dotar la capacidad al administrador de moldear la red según sus necesidades.

- El emulador Mininet, es una herramienta ágil y de gran utilidad en cuanto a la creación y puesta en marcha de topologías SDN, se puede ejecutar cualquier tipo de aplicación o comando de sistema desde cualquiera de los dispositivos creados, obteniendo resultados tan cercanos a la realidad.

- De las múltiples alternativas de software que existen para la implementación de un controlador SDN, se escogió POX un controlador OpenFlow de código abierto basado en Python de fácil entendimiento con mucha documentación para su estudio y comprensión pero lo más importante su versatilidad al momento de desarrollar componentes personalizados, como es el caso del componente firewall_l2 que se configuro para este proyecto, la función del componente es la de leer parámetros de configuración de un archivo y enviar reglas basadas en estos parámetros a los conmutadores mediante mensajes de modificación de flujo.

- La configuración y aplicación de las políticas de control para el flujo de tráfico entre dispositivos, es más fácil realizarlo en un firewall basado en SDN que en un firewall tradicional, alcanzando de esta manera la escalabilidad y flexibilidad que actualmente un entorno de red lo requiere.

- Los resultados obtenidos en las pruebas realizadas son los esperados, se aprecia como los paquetes enviados a los hosts cuya MAC concuerda con alguna política de restricción son dropeados, mientras que los demás siguen su flujo normal hasta llegar a su destino, adicional también se pudo constatar que el primer paquete ICMP que se envía de un host a otro tiene un tiempo de respuesta alto en comparación a los siguientes y esto es debido a que dicho paquete tiene que viajar hasta el controlador SDN para que este indique a los switches el flujo con el que deben continuar los paquetes ICMP.

- El emulador Mininet es una herramienta de fácil instalación y muy práctica para desarrollar y experimentar con entornos SDN, por lo que es recomendable utilizarla para familiarizarse con conceptos y procedimientos propios de OpenFlow y SDN, ganando de esta manera experticia para el desarrollo de proyectos más complejos en un futuro.

- El tener dos controladores SDN que trabajen en una arquitectura redundante o failover, sería muy aconsejable ya que al momento de presentarse una falla en alguno de ellos no afectaría el normal comportamiento de la red, así como tampoco se comprometería la información que está viajando en ella.

Bibliografía

[1] C.A., T. E. (1 de Julio de 2012). *Tecnosat Electrónica C.A.: Las redes definidas por software (SDN) se convertirán en nueva norma universal*. Obtenido de http://tecnosatelectronica.blogspot.com/2012/07/las-redes-definidas-por-software-sdn-se.html

[2] Centeno, A. G. (Septiembre de 2014). Controladores SDN, elementos para su selección y evaluación. *Revista Telem@tica, 13*(3), 10-20.

[3] Cisco. (s.f.). *Software Defined Networking (SDN) - Overview*. Obtenido de http://www.cisco.com/c/en/us/solutions/software-defined-networking/overview.html

[4]Cloud, S. (21 de Septiembre de 2016). *¿Sabes que son las Redes definidas por software?* Obtenido de http://www.somoscloud.com/redes-definidas-software/

[5] Enterprise, H. P. (2016). *Software Defined Networking - SDN Management Software and Solutions*. Obtenido de https://www.hpe.com/us/en/networking/sdn.html

[6] Foundation, O. N. (2014). *SDN architecture*. Obtenido de https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_SDN_ARCH_1.0_06062014.pdf

[7] Foundation, O. N. (2016). *Home - Open Networking Foundation*. Obtenido de https://www.opennetworking.org/sdn-resources/openflow

[8] GitHub. (2016). *Mininet - Emulator for rapid prototyping of Software Defined Networks*. Obtenido de Mininet: https://github.com/mininet/mininet

[9] Goransson, P., Black, C., & Culver, T. (2016). *Software Defined Networks: A Comprehensive Approach.* Cambridge, United States: Morgan Kaufmann.

[10] Herrera, R. F. (2 de Septiembre de 2015). *Principales Caso de uso de SDN*. Obtenido de http://aunclicdelastic.blogthinkbig.com/casos-de-uso-de-sdn-la-busqueda-continua/

[11] Millán, R. (2006 - 2016). *SDN: el futuro de las redes inteligentes*. Obtenido de http://www.ramonmillan.com/tutoriales/sdnredesinteligentes.php

[12] Mininet. (2016). *Mininet An Instant Virtual Network on your Laptop (or other PC)*. Obtenido de http://mininet.org/

[13] Nadeau, T. D., & Gray, K. (2013). *SDN: Software Defined Networks: An Authoritative Review of Network Programmability Technologies.* O'Reilly Media, Inc.

[14] Networks, J. (s.f.). *Juniper Networks Software Defined Networking Solution - Juniper Networks*. Obtenido de https://www.juniper.net/us/en/solutions/sdn/

[15] Peralta, R. M. (s.f.). *Emulación de SDNs usando Mininet*. Obtenido de http://www.profesores.elo.utfsm.cl/~agv/elo323/2s14/projects/reports/RodrigoManriquez/RodrigoManriquez.pdf

[16] Python. (2001 - 2016). *The official home of the Python Programming Language*. Obtenido de https://www.python.org/

[17] RAMIREZ, J. L. (2015). *Repositorio.utp.edu.co*. Obtenido de http://repositorio.utp.edu.co/dspace/bitstream/handle/11059/5713/00678H493%20_Anexo.pdf?sequence=2

[18] Udacity. (2011 -2016). *Assignment 7: SDN and Firewalls - Udacity*. Obtenido de https://www.udacity.com/wiki/cn/assignment7-sdn-firewall

[19] Vargas, W. (2016). *Emulación de una red definida por software utilizando MiniNet*. Obtenido de https://www.academia.edu/5730624/Emulaci%C3%B3n_de_una_red_definida_por_software_utilizando_MiniNet

[20] Vogel, J. D. (19 de Marzo de 2015). *Using Software Defined Networking To Solve Missed Firewall Architecture In Legacy Networks*. Obtenido de http://ir.library.illinoisstate.edu/cgi/viewcontent.cgi?article=1365&context=etd