UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACION TÉCNICA PARA EL DESARROLLO

MAESTRÍA EN TELECOMUNICACIONES

TEMA:

SIMULACIÓN DE UNA RED DEFINIDA POR SOFTWARE (SDN) PARA EL CONTROL DE ACCESO DE LOS ELEMENTOS DE LA RED A NIVEL DE CAPA 2

AUTOR:

CARRILLO RODAS CARLOS ALBERTO

Previo a la obtención del grado de
Magister en Telecomunicaciones

TUTOR:

ING. MANUEL ROMERO PAZ, M.SC.

Guayaquil, 29 de junio del 2020

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

SISTEMA DE POSGRADO

MAESTRÍA EN TELECOMUNICACIONES

CERTIFICACIÓN

Certificamos que el presente trabajo fue realizado en su totalidad por **Carlos Alberto Carrillo Rodas** como requerimiento parcial para la obtención del Título de Magíster en Telecomunicaciones.

TUTOR

MSc. Manuel Romero Paz

DIRECTOR DEL PROGRAMA

MSc. Manuel Romero Paz

Guayaquil, 29 de junio del 2020

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

SISTEMA DE POSGRADO

MAESTRÍA EN TELECOMUNICACIONES

DECLARACIÓN DE RESPONSABILIDAD

Yo, **Carlos Alberto Carrillo Rodas**

DECLARO QUE:

La Tesis "**Simulación de una red definida por software (SDN) para el control de acceso de los elementos de la red a nivel de capa 2**", previo a la obtención del Título de **Magíster en Telecomunicaciones**, ha sido desarrollado respetando derechos intelectuales de terceros conforme las citas que constan en el documento, cuyas fuentes se incorporan en las referencias o bibliografías. Consecuentemente este trabajo es de mi total autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance del Trabajo de Titulación referido.

Guayaquil, 29 de junio del 2020

EL AUTOR

Carlos Alberto Carrillo Rodas

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

SISTEMA DE POSGRADO

MAESTRÍA EN TELECOMUNICACIONES

AUTORIZACIÓN

Yo, **Carlos Alberto Carrillo Rodas**

Autorizo a la Universidad Católica de Santiago de Guayaquil a la **publicación**, en la biblioteca de la institución de la Tesis de Maestría titulada, **"Simulación de una red definida por software (SDN) para el control de acceso de los elementos de la red a nivel de capa 2"**, cuyo contenido, ideas y criterios son de mi exclusiva responsabilidad y total autoría.

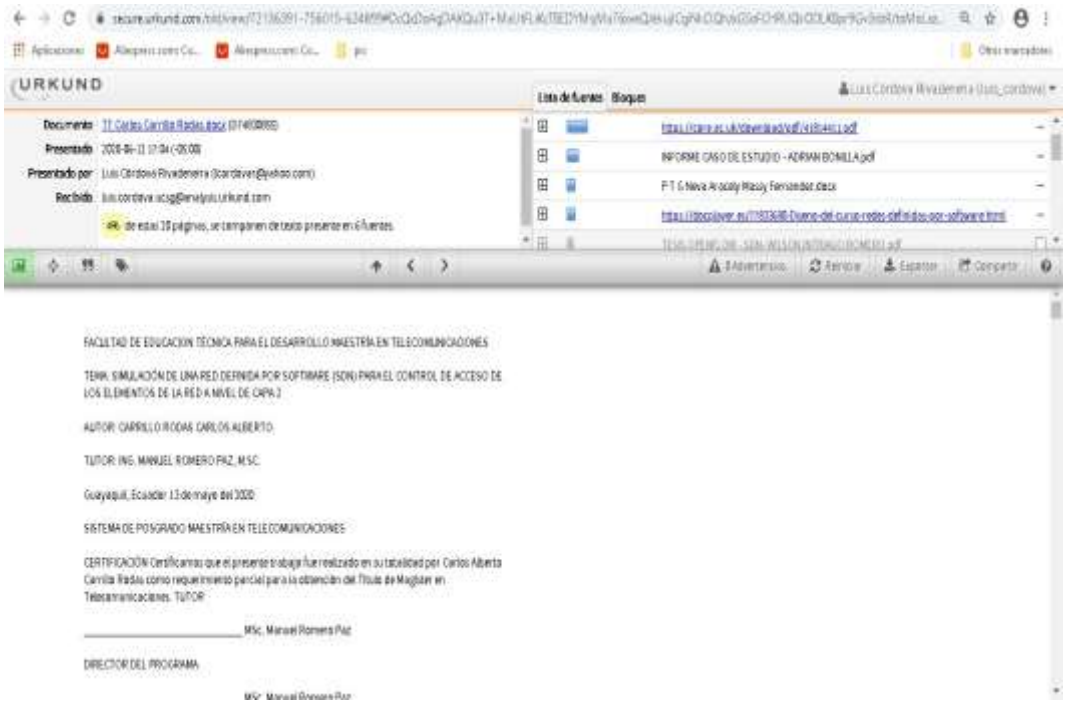
Guayaquil, 29 de junio del 2020

EL AUTOR

_____

Carlos Alberto Carrillo Rodas

REPORTE URKUND



DEDICATORIA

Dedico este estudio primero a Dios, a mis padres que siempre me han apoyado en cada etapa de mi vida, a mi esposa Edith y mi hija Arianna que gracias a ellas tuve la motivación necesaria para dar cumplimiento a este objetivo y a mis hermanos Fabian, Sandra y Jhoana que siempre están pendientes de mí.

AGRADECIMIENTO

Agradezco primero a Dios por bendecirme cada día, por darme la sabiduría de llegar hasta el final, a mis padres y hermanos por siempre apoyarme, a mi esposa y mi hija por motivarme a que finalice esta etapa y cumpla mi meta y a mis suegros y cuñados que siempre han estado pendientes de que culmine este objetivo.

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

SISTEMA DE POSGRADO

MAESTRÍA EN TELECOMUNICACIONES

TRIBUNAL DE SUSTENTACIÓN

f. 

MSc. Manuel Romero Paz

TUTOR

f. 

MSc. Manuel Romero Paz

DIRECTOR DEL PROGRAMA

f. 

MSc. Luis Córdova Rivadeneira

REVISOR

f. 

MSc. Edgar Quezada Calle

REVISOR

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El siguiente caso de estudio se divide en cuatro capítulos: en el primero se describe la investigación a realizar como son los objetivos planteados, el planteamiento del problema y la hipótesis a analizar, en el siguiente capítulo se realiza el estudio del arte de la redes SDN, protocolo OpenFlow, así como de los tipos de controladores SDN, parte teórica fundamental para el diseño de la red SDN la cual está definida en el capítulo tres, así como todas las herramientas necesarias de hardware y de software para comprobar su funcionamiento, como último capítulo se tiene la simulación, pruebas y análisis de la misma en una red definida por software para el control de acceso de los elementos de la red a nivel de capa 2. El tipo de investigación a desarrollar es exploratoria, por la recolección de conocimientos previos sobre redes SDN, descriptiva porque se va caracterizar el objeto de estudio, aplicada ya que se utilizarán los conocimientos obtenidos en la investigación para la simulación de una red SDN para el control de acceso de los elementos de la red a nivel de capa 2, y transversal ya que los resultados se los realizará en un momento y tiempo definido. La metodología a aplicarse es el método deductivo - inductivo para la recopilación y caracterización de la información, analítico - sintético para descomponer el objetivo principal en varios objetivos específicos, para al finalizar recopilar la información y por medio de la síntesis encontrar la solución al problema, y con un enfoque cuantitativo con el cual mediante pruebas se desea obtener información de la funcionalidad de una de sus características.

Palabras Clave: Redes SDN, OpenFlow, MiniNet, Floodlight, Control de Acceso, Seguridad de la red.

ABSTRACT

The following case study is divided into four chapters: the first describes the research to be carried out, such as the objectives set, the problem statement and the hypothesis to be analyzed. In the next chapter, the study of the art of SDN networks, the Open Flow protocol, as well as the types of controllers SDN is carried out, fundamental theoretical part for the design of the SDN network which is defined in chapter three, as well as all the necessary hardware and software tools to check its operation, as the last chapter there is the simulation, testing and analysis of the same in a network defined by software for access control of network elements at the layer 2 level. The type of research to be carried out is exploratory, for the collection of previous knowledge on SDN networks, descriptive because the object will be characterized study, applied since the knowledge obtained in the research will be used for the simulation of an SDN network for access control of the elements of the network at the level of layer 2, and cross-sectional since the results will be carried out at a defined moment and time. The methodology to be applied is the deductive - inductive method for the collection and characterization of the information, analytical - synthetic to decompose the main objective into several specific objectives, to finally collect the information and through synthesis find the solution to the problem, and with a quantitative approach with which through tests you want to obtain information on the functionality of one of its characteristics.

Keywords: SDN Networks, OpenFlow, MiniNet, Floodlight, Access Control, Network Security

1. CAPÍTULO I. DESCRIPCIÓN DE LA INVESTIGACIÓN

En el capítulo I se indicará la descripción de la investigación como la introducción, antecedentes, justificación, definición del problema, objetivos, hipótesis y la metodología a utilizar.

1.1. Introducción.

A nivel mundial las exigencias de la sociedad, los avances científicos y tecnológicos han permitido ir evolucionando las telecomunicaciones, además las limitaciones que las redes actuales presentan ante requerimientos de los usuarios, han provocado que en el campo de las redes se busque una nueva arquitectura que cumpla con sus necesidades, a esta nueva arquitectura se la denomina Red Definida por Software (SDN, Software-Defined Networking).

Las ventajas de las redes SDN son: reducir costo de equipamiento y de operación, control de la red con mayor flexibilidad y la introducción de nuevos servicios con mayor facilidad.

Posee una arquitectura de red que separa de forma efectiva los planos de control y de datos, por lo que su trabajo no requiere un equipo, sino un controlador de programas de computador (Bonilla, 2016).

En este proyecto se pretende presentar una simulación de red SDN para el control de acceso a nivel de capa 2, por lo cual se realizará inicialmente un estudio teórico de estas redes, donde se analizará el funcionamiento y sus características para posteriormente desarrollar una simulación de la red con su respectivo controlador, el cual por medio del protocolo OpenFlow decidirán si un paquete de datos se bloqueará o transmitirá dentro de la red, finalmente mediante la herramienta MiniNet se emulará los elementos de la red para experimentar su funcionalidad.

1.2. Antecedentes.

En la actualidad el internet y las aplicaciones de las redes de computadoras han crecido considerablemente, un gran flujo de datos existe a través de internet y de redes privadas, lo que ha provocado que el control y la configuración de los dispositivos de la red sean más complejos. Estos equipos en la actualidad ejecutan software de control complicado y en su gran mayoría son distribuidos por lo que suele ser cerrado y propietario de cada marca como son: enrutadores, conmutadores, cortafuegos, DNS (Domain Name System), entre otros.

Las redes tradicionales siguen funcionando a nivel de protocolos individuales, mecanismos e interfaces de configuración por lo que se ha incrementado la complejidad y los costos de operación de la red.

Por lo antes mencionado, en estos últimos años se ha innovado el diseño y gestión de las redes buscando que estas sean más programables, por eso la aparición de un nuevo tipo de redes denominadas SDN, las cuales tienen dos características primordiales, separar el plano de control del de datos y consolidar el primero para que un sistema de software controle los múltiples elementos del plano de datos, en el Ecuador aún no se tiene una red SDN implementada, pero en varias universidades se han realizado estudios de las características y beneficios que este tipo de redes podrán brindar en el futuro.

1.3. Justificación.

Las empresas hoy en día dependen en gran medida de las redes informáticas, además reconocen que la seguridad de la información es prioritaria para el correcto desarrollo de ésta, por lo que uno de los aspectos principales para no verse comprometidos por amenazas de seguridad es la utilización de firewalls, la cual muchas empresas obvian por su alto costo de adquisición o por su compleja programación, dejando de esta manera vulnerable a la red a posibles amenazas de seguridad.

Las redes SDN están siendo estudiadas en los últimos años como una sustitución de la arquitectura tradicional, su principal ventaja es la reducción de costos de equipamiento y operación, por lo cual es necesario realizar un estudio para conocer las características de seguridad que estas redes pueden proponer, reduciendo la complejidad de las arquitecturas de seguridad, dando dinamismo a la red y que sea capaz de adaptarse fácilmente a las necesidades de los usuarios, así como del negocio.

1.4. Planteamiento del Problema.

La tecnología en el mundo de las redes de telecomunicaciones es cambiante de forma constante, por tal razón se ve la necesidad de estudiar nuevas formas con lo que la gestión de las redes no se remedia con equipos, sino mediante un controlador de software, con redes SDN.

1.5. Definición del Problema

La necesidad de estudiar y comprender el funcionamiento, características y beneficios que las redes definidas por software (SDN) pueden brindar a una red tradicional.

1.6. Objetivos.

1.6.1. Objetivo General.

Simular una red definida por software (SDN) con la funcionalidad de controlar el flujo de datos a nivel de capa 2.

1.6.2. Objetivos Específicos.

- Investigar el estado del arte de la tecnología SDN.
- Caracterizar los controladores existentes en las redes SDN.
- Simular la red SDN con el programa MiniNet.
- Configurar la capa de control de los equipos, con un controlador SDN.

1.7. Hipótesis.

Si se simula una red definida por software (SDN) para el control de flujo de datos, se evaluará una de las funcionalidades de este tipo de red, para en un futuro poder ser utilizada como solución de las redes LAN (Local Area Network) en el campo de seguridad.

1.8. Metodología.

El tipo de investigación a desarrollar es exploratoria, por la recolección de conocimientos previos sobre redes SDN, descriptiva porque se va caracterizar el objeto de estudio, aplicada ya que se utilizarán los conocimientos obtenidos en la investigación para la simulación de una red SDN para el control de acceso de los elementos de la red a nivel de capa 2, y transversal ya que los resultados se los realizará en un momento y tiempo definido.

La metodología que se desarrollará es el método deductivo - inductivo para la recopilación y caracterización de la información, analítico - sintético ya que se va a descomponer el objetivo principal en varios objetivos específicos, para al finalizar recopilar toda esta información y por medio de la síntesis encontrar la solución al problema, y con un enfoque cuantitativo con el cual mediante pruebas se desea obtener información de la funcionalidad de una de sus características.

2. CAPÍTULO II. MARCO TEÓRICO

En este capítulo se describirá los fundamentos teóricos necesarios para el correcto entendimiento de las redes SDN, fundamentales para el desarrollo de la simulación planteada.

2.1. Redes SDN

Estas redes aparecen en 1990, cuando se agregan funciones programables a la red, la separación del plano de control y de datos se incluye en el periodo 2001 – 2007, la integración de los API (Application Programming Interface) OpenFlow como una interfaz abierta, donde se presentan diferentes maneras de separar los planos antes mencionados, convirtiéndola en una red escalable, se desarrolla en los años del 2007 – 2010.

La tecnología SDN se ha venido desarrollando para ofrecer soluciones configurables, económicas, dinámica, escalable y adaptable a cualquier red, lo que las redes tradicionales hoy en día ya no pueden ofrecer (Bonilla, 2016). Con una red SDN el administrador puede modificar el comportamiento de la red, ya que el control se encuentra centralizado en un equipo denominado controlador, por lo que ya no es necesario como en las redes tradicionales ir configurando en cada equipo de la red.

2.1.1. Características de la red SDN

Las principales características de una red SDN son las siguientes: **directamente programable,** mediante programación se tiene el control de la red; **gestión de forma centralizada,** mediante un único switch lógico basado en controladores SDN se mantiene una visión global de la red; **ágil,** el flujo de tráfico puede ser administrado de forma dinámica; **basado en estándares abiertos y de proveedor neutral,** las instrucciones no son controladas por múltiples dispositivos y protocolos específicos de los fabricantes sino por los controladores SDN de una manera en que se simplifica el diseño de la red y las operaciones; **programación**

configurada, al no depender de software propietario, los administradores de la red SDN pueden configurar, asegurar, optimizar y administrar los recursos de la misma rápidamente a través de programas SDN.

2.1.2. Componentes de la red SDN

Las redes SDN se componen por tres capas: Infraestructura, Control y Aplicación (figura 2.1).

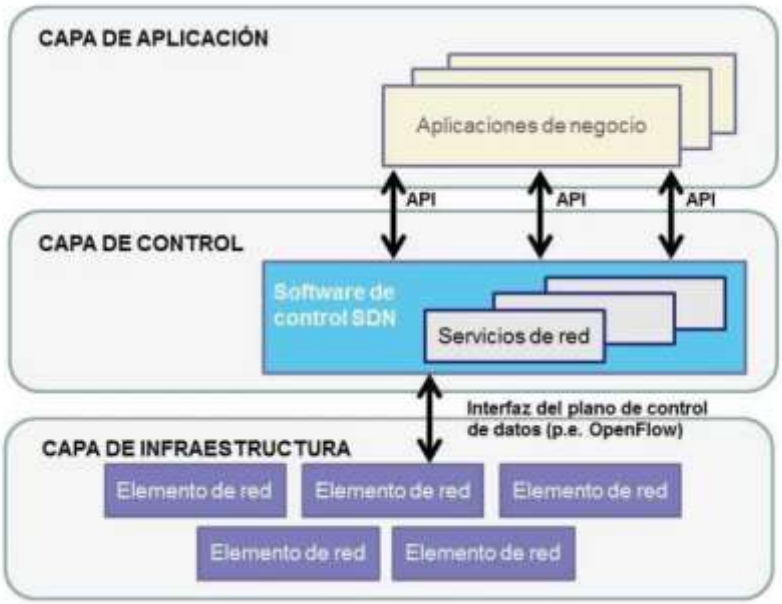


FIGURA 2.1. Arquitectura de SDN.
FUENTE: (Millán, 2014)

La Capa de Infraestructura, es aquella compuesta por los equipos routers, switch, etc., encargados del enrutamiento y conmutación de paquetes, esta capa por medio de la API hacia abajo (southbound) facilita un acceso abierto programable.

La Capa de Control, en la cual se encuentra el controlador SDN, software encargado de controlar y configurar los nodos de la red. Además, es el ente responsable de tomar decisiones y seleccionar el mejor camino para un correcto flujo de tráfico, asegurando los aspectos necesarios para la interoperabilidad de la red, esta es la capa intermedia, la cual por medio de las interfaces API hacia abajo (southbound) se conecta con la capa de infraestructura y por la API hacia arriba (northbound) se conecta con la capa de aplicación.

La Capa de Aplicación, es donde se encuentran las aplicaciones de alto nivel solicitadas por los usuarios, estas solicitudes mediante la interfaz API hacia arriba (northbound) son enviadas y receptadas por el controlador, lo que permite simplificar y automatizar las tareas de configuración (Millán, 2014).

2.1.3. Ventajas de la red SDN

Gracias a la virtualización se puede aumentar la seguridad y adaptarse a los cambios de la red en las redes SDN. Las ventajas de la red SDN son:

- Las redes SDN son más flexibles y ahorran costos: permiten adaptarse a diferentes necesidades de la red sin requerir la compra de nuevos nodos, gracias a la virtualización de la infraestructura de la red.
- Las redes definidas por software tienen una gestión centralizada y simple: son administradas de manera sencilla y desde un único lugar, lo que permite adaptar las necesidades de manera simplificada y rápida. Debido a la abstracción de los planos de control y de datos, las redes SDN permiten equilibrar la carga y distribuir el tráfico, mejorando el rendimiento y además por su configuración e instalación automática, se reduce costos operativos.
- La automatización: permite que la red SDN tenga un entorno predecible y consistente, con una escalabilidad que facilita acomodar picos y valles de tráfico, en función de las necesidades surgidas en cada momento.
- Aumentan la seguridad: la red SDN es más robusta y mejora la seguridad gracias a la automatización de procesos, permitiendo que los problemas sean resueltos fácilmente desde un plano de control centralizado, proporcionando una seguridad de gran precisión a las aplicaciones y puntos finales.
- Una infraestructura SDN aprovecha el Cloud Computing: la computación en la nube es una de las grandes revoluciones en la industria de la informática y de Internet. A medida que las empresas se trasladan a la nube, su infraestructura debe ser virtualizada y administrada centralmente para crear nuevos servicios en la parte superior de la red (Puntinformatic, s.f.).

2.1.4. Diferencia entre la red tradicional vs red SDN

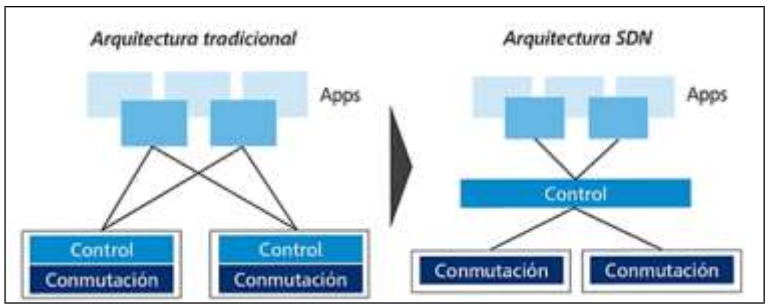


FIGURA 2.2. Red tradicional vs Red SDN.
FUENTE: (Ríos, 2016)

Debido a los cambios tecnológicos existentes en las redes es posible decir que las redes SDN hoy en día son una tendencia, en la tabla 2.1 se muestras las diferencias entre las redes tradicionales y las redes SDN.

TABLA 2.1. Diferencia entre Red Tradicional vs Red SDN.

Redes Tradicionales	Redes SDN
Relativamente estática y difícil de aplicar políticas	Dinámica
Incapaz de ser escalable	Escalable
Las tablas de flujo están cerradas en los dispositivos	Tablas de enrutamiento abiertas
Configuración descentralizada	Configuración centralizada
Interfaces propietarias	API's bien definidas y abiertas
Envió de paquetes basados en coincidencias de la tabla	Formato y acciones de las tablas claramente especificadas
Altos costos operativos y financieros	Reducción de costos operativos y financieros

FUENTE: Elaborado por el Autor.

2.2. Protocolo OpenFlow

Es establecido por la ONF (Open Networking Foundation), por medio de grupos de trabajo, pruebas de interoperabilidad y otras actividades del protocolo están encargadas de la estandarización de la misma. OpenFlow en una arquitectura, SDN es la interfaz de comunicaciones estándar entre los planos de control y de datos.

2.2.1. Funcionamiento del Protocolo OpenFlow

Los controladores en OpenFlow son los encargados de tomar la decisión del reenvió de paquetes. Los switches están unidos al controlador mediante una aplicación que se ejecuta en la interfaz facilitando la configuración y administración del flujo de la red.

Para definir un flujo dentro de la red se pueden considerar los siguientes parámetros: los terminales de recepción de los paquetes y los de inicio, el tag VLAN (Virtual Local Area Network), el destino y otras particularidades de los paquetes (Intriago, 2017).

Si un paquete no coincide con las entradas de la tabla, el controlador es el encargado de definir un nuevo flujo para ese paquete y crear una o más entradas en la tabla, luego ésta es enviada al switch para que sea ingresado a la tabla de flujo y finalmente procesado (Intriago, 2017).

En la figura 2.3 se muestra claramente cómo se ubica el protocolo OpenFlow entre el switch y el controlador y en software el canal de enlace con el protocolo SSL (Secure Sockets Layer) y a nivel de hardware la tabla de flujos (Intriago, 2017), (electiva redes avanzadas, 2015).

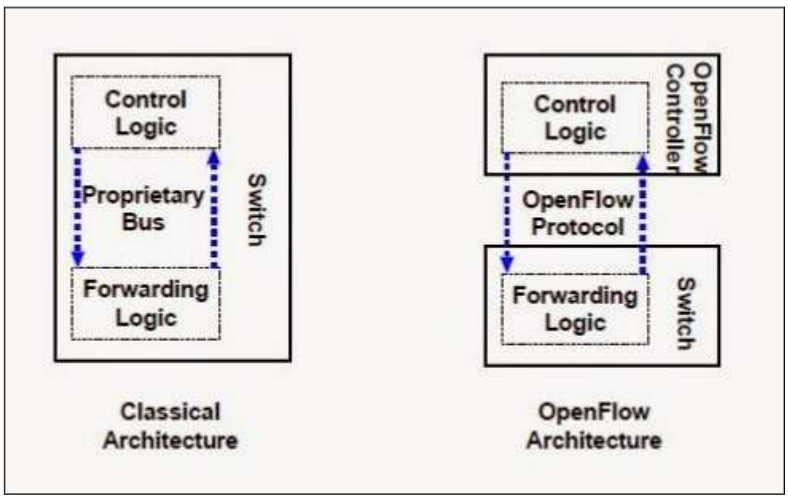


FIGURA 2.3. Arquitectura OpenFlow.
FUENTE: (electiva redes avanzadas, 2015)

2.2.2. Componentes del Conmutador OpenFlow

Estos resisten muchas situaciones, si bien con rasgos similares. Su construcción se puede observar en la figura 2.4 (Marín, 2016), (Marín, Y., 2016).

FIGURA 2.4. Componentes del conmutador OpenFlow.
FUENTE: (Marín, Y., 2016)**,** (Marín, 2016)

Tabla de flujos: Con una acción asociada a cada entrada de la tabla, indicando al switch como debe procesar ese flujo.
Canal seguro: que conecte el switch a un proceso de control remoto (controlador), que permita el envío de comandos y paquetes entre el conmutador y el controlador usando el protocolo OpenFlow.
Controlador: Un controlador añade y elimina entradas en la tabla de flujos (Marín, Y., 2016) (Marín, 2016).

2.2.3. Versiones del Protocolo OpenFlow

El protocolo OpenFlow aparece en diciembre del 2009 con su versión 1.0 y en el transcurso de los años se ha ido actualizando para cumplir con las nuevas necesidades de las redes como la escalabilidad y administración de red, a continuación, se da un resumen de las versiones del protocolo OpenFlow indicando sus características más relevantes (Chang & Lin, 2015).

OpenFlow 1.0

- Aparece en diciembre de 2009
- Una sola tabla de flujo con solo tres componentes: campos de cabecera, contadores y acciones.
- Solo existen 12 campos de coincidencia en los campos de cabecera.
- Poca flexibilidad de administración sobre la red.
- Cuenta con una sola tabla de flujo lo que produce el problema de explosión de entradas.

OpenFlow 1.1

- Aparece en febrero de 2011
- Introducción de múltiples tablas de flujo y grupos de tablas, lo que permite un mejor manejo de los paquetes en el flujo de la red.
- Los campos de cabecera y las acciones son renombradas a campos de coincidencia e instrucciones respectivamente.

OpenFlow 1.2

- Aparece en diciembre de 2011
- Introducción de la estructura TLV(Type-Lenght-Value) que permite añadir más campos de comparación de una forma más modular. Esta funcionalidad se denomina OpenFlow Extensible Match (OXM).
- Soporte básico para IPv6 a través de OXM.
- Se añade el mecanismo de cambio de rol de controlador, lo que permite utilizar más de un controlador en una red y disminuir el riesgo del único punto de falla.

OpenFlow 1.3

- Aparece en abril de 2012
- Se mejora el soporte para QoS con la inclusión de la tabla de medición. Cada entrada en la tabla de medición contiene una lista de bandas de medidor, las cuales especifican el comportamiento de la red ante cierto tipo de tráfico.

- Se extiende la tabla de flujo con una tabla de entradas perdidas, esto permite obtener un mejor comportamiento de la red al momento que un paquete no coincida con las entradas de las tablas de flujo y que fueron descartados

OpenFlow 1.4

- Aparece en agosto de 2013
- Introducción de sincronización de tablas la cual puede ser de forma bidireccional o unidireccional
- Se añade una nueva característica denominada empaquetamiento, que permite preparar y validar varios mensajes OpenFlow que serán aplicados por múltiples switches.

OpenFlow 1.5

- Aparece en enero de 2015
- Introduce la calendarización de paquetes mediante la inclusión de tiempo de ejecución como propiedad del paquete. Un conmutador que recibió un paquete programado aplicará el mensaje lo más cerca posible del tiempo de ejecución. Esto fortalece aún más la sincronización entre múltiples switches.

2.3. Controladores SDN

En una red SDN el núcleo y componente más importante es el controlador, encargado de traducir las necesidades de la capa de aplicación a los elementos de la red SDN, por lo que se puede decir que es el cerebro de ella y quien dirige los flujos de la red y sus elementos solo obedecen sus reglas.

En la actualidad se tienen controladores de código abierto como: Beacon, POX, Floodlight, OpenIRIS, etc., solo se distinguen por la programación para precisar las normas y estrategias y la plataforma en que operen.

2.3.1. POX (Python)

Considerado como una nueva versión de NOX basado en Python, es un controlador OpenFlow para satisfacer las exigencias de la red SDN que permite un rápido desarrollo y creación de prototipos de software de control de la red (Martínez, 2015).

POX por su época de desarrollo no tiene interfaz web.

Entre los parámetros principales de POX están (Bonilla, 2016):

- Interfaz OpenFlow de Python.
- Apoyo de plataformas Linux, Mac OS y Windows.
- Con soporte especial para Open VSwitch.

2.3.2. OpenIRIS (Java)

OpenIRIS es una versión de código abierto de IRIS, es un controlador SDN basado en OpenFlow diseñado para resolver problemas de escalabilidad y disponibilidad de SDN (IRIS, 2020).
Sus características son:

- Soporta OpenFlow 1.0.1 ~ 1.3.2.
- API de OpenFlow basada en Loxigen.
- Implementación pura basada en Java.
- Admite alrededor de 500 conexiones simultáneas a conmutadores con Commodity HW.
- Sigue la misma política de licencias que Floodlight (licencia Apache-2.0).
- Proporciona aprendizaje MAC, descubrimiento de enlaces, gestión de topología, reenvío, administrador de dispositivos, firewall, conmutación por error de red, impulsor de flujo estático y módulos de administrador de estado.

Brinda una interface web para representar elementos de red, enlaces y otras particularidades que la precisan como se muestra en la figura 2.5.

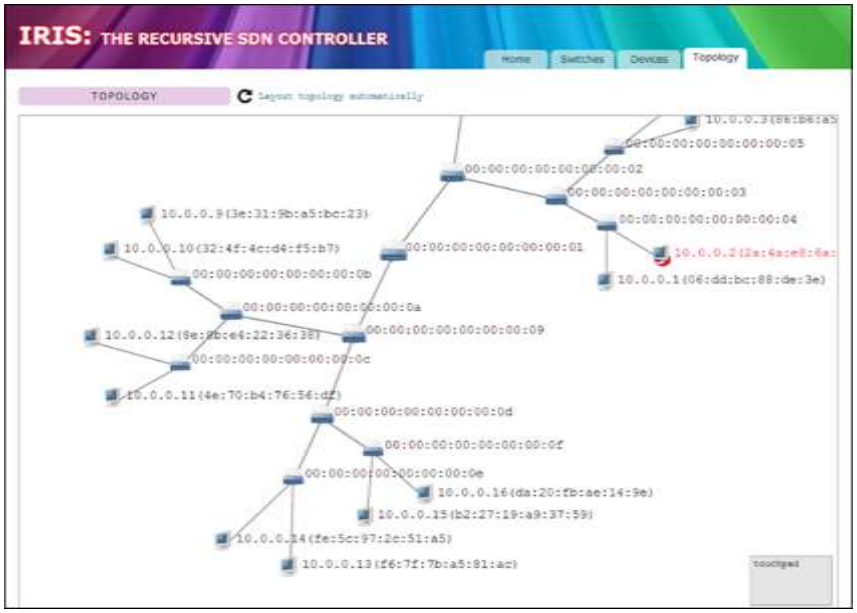


FIGURA 2.5. Interfaz web del controlador OpenIRIS.
FUENTE: (IRIS, 2020)

2.3.3. Floodlight (Java)

Floodlight es un controlador OpenFlow basado en Java de clase empresarial con licencia de Apache.

Está diseñado para funcionar con el creciente número de switches, routers, switch virtuales y puntos de acceso que admite el estándar OpenFlow (Project Floodlight, 2020).

Sus características principales son (Mendoza, 2016):

- Ofrece un sistema de carga de módulos que simplifica la ampliación y mejora.
- Configurable con observancias mínimas.
- Permite una gran progresión de switches OpenFlow virtuales y físicos.
- Operan redes mixtas OpenFlow y no OpenFlow: para variadas "islas" de switches de equipos OpenFlow.
- Planteado para elevado rendimiento: es multiproceso.

Muestra una interface web para observar el estudio usado y sus elementos (switch, hosts, enlaces) como se muestra en la figura 2.6.

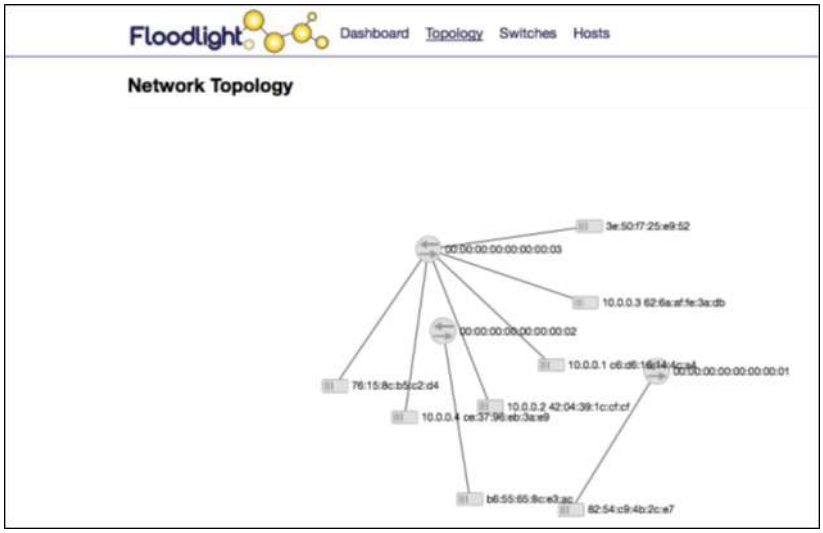


FIGURA 2.6. Interfaz web del controlador Floodlight.
FUENTE: (Project Floodlight, 2020)

2.4. MININET

Emulador que genera una red de hosts virtuales, switches, controladores y conexiones. Estos hosts operan un software Linux y sus switches admiten OpenFlow para un enrutamiento individualizado muy flexible en una SDN.

MiniNet permite investigación, desarrollo, aprendizaje, creación de prototipos, pruebas, depuración y otras labores que podrían usar una red experimental completa en una laptop o PC.

2.4.1. Características de MININET

MiniNet combina muchas de las mejores características de emuladores, bancos de pruebas de hardware y simuladores (Mininet Team, 2018).

- Incluye un banco de pruebas de red sencillo y barato para desplegar aplicaciones OpenFlow.
- Varios desarrolladores pueden trabajar independientemente en el mismo estudio.
- Permite pruebas de retroceso a nivel de sistema, que son repetibles y se empaquetan fácilmente.
- Permite pruebas complejas de topología, sin la necesidad de conectar una red física.

- Incluye una CLI (Command-Line Interface) que reconoce la topología y OpenFlow, para depurar o ejecutar pruebas en toda la red.
- Admite topologías personalizadas arbitrarias e incluye un conjunto básico de topologías parametrizadas.
- Es usable fuera de la caja sin programación.
- Proporciona una API Python sencilla y extensible para la creación y experimentación de redes.

En comparación con los enfoques basados en la virtualización del sistema completo, MiniNet:

- Arranca más rápido: segundos en lugar de minutos.
- Escalas más grandes: cientos de hosts y switches frente a un solo dígito.
- Proporciona más ancho de banda: por lo general, 2 Gbps de ancho de banda total en hardware.
- Se instala fácilmente: hay disponible una máquina virtual pre-empaquetada que se ejecuta en VMware o VirtualBox para Mac / Win / Linux con las herramientas OpenFlow v1.0 ya instaladas.

En comparación con los bancos de pruebas de hardware, MiniNet

- Es económico y siempre está disponible.
- Es rápidamente reconfigurable y reiniciable.

En comparación con los simuladores, MiniNet

- Ejecuta código real sin modificar, incluido el código de la aplicación, el código del kernel del sistema operativo y el código del plano de control.
- Se conecta fácilmente a redes reales.
- Ofrece rendimiento interactivo: puede escribir en él.

3. CAPÍTULO III. DISEÑO DE LA RED SDN

En este capítulo III se desea dar a conocer los elementos de hardware y software necesarios para el diseño y simulación de la red SDN, tomando en cuenta lo expuesto en el capítulo anterior.

3.1. Elementos de Hardware

La PC portátil donde se va instalar el emulador MiniNet tiene las siguientes características (Bonilla, 2016):

- Marca y Modelo: ASUS TP301UA.
- Procesador: Intel Core (TM) i7 2.59 Ghz.
- Memoria RAM: 8 GB.
- Sistema Operativo: Windows 10.

3.2. Elementos de Software

Los elementos de software utilizados en este estudio para la simulación de la red SDN son: Oracle VM VirtualBox, Floodlight, MININE y SecureCRT.

3.2.2. Instalación del controlador Floodlight.

Luego de realizar la caracterización de los controladores en el capítulo anterior, se ha optado por utilizar el controlador Floodlight porque está basado en Java, maneja una interfaz web para la visualización de la topología, además puede soportar una gran gama de switches y es de fácil configuración.

Para su instalación se descarga la máquina virtual de Floodlight que viene con el entorno configurado y listo para su correcto funcionamiento, en el siguiente link:

https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/8650780/Floodlight+VM.

El archivo descargado viene en formato comprimido, y contiene un archivo con extensión *.vmdk, que es el disco de datos virtual.

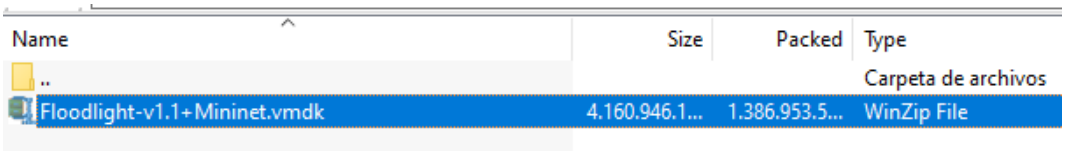


FIGURA 3.1. Archivos de instalación del controlador Floodlight.
FUENTE: Elaborado por el Autor

3.2.3. Instalación en Oracle VM VirtualBox

Para continuar se debe ejecutar la máquina virtual descargada, por medio del software de virtualización Oracle VM VirtualBox el que se podrá descargar del siguiente link:

https://www.virtualbox.org/wiki/Downloads.

A continuación, se describe los pasos a seguir para una correcta configuración del software Oracle VM VirtualBox y ejecución de la máquina virtual Floodlight (Bonilla, 2016):

1) Ejecutar el archivo VirtualBox-6.0.14-133895-Win.exe descargado y seguir los pasos de instalación.

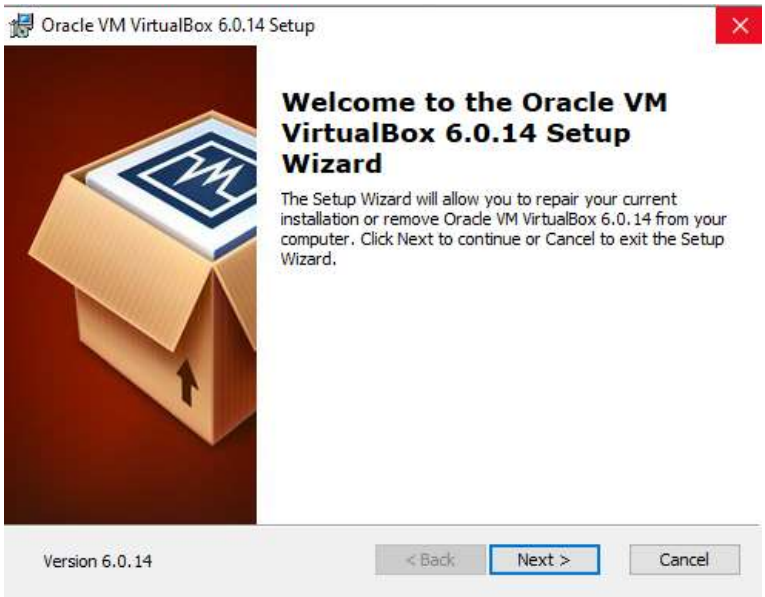


FIGURA 3.2. Instalación de la máquina virtual Oracle VM VirtualBox
FUENTE: Elaborado por el Autor

2) Una vez instalado Oracle VM VirtualBox, se debe importar la máquina virtual seleccionando la opción “Importar servicios virtualizados” y posteriormente escogiendo el archivo *. vmdk de donde se lo tenga descargado (Bonilla, 2016).

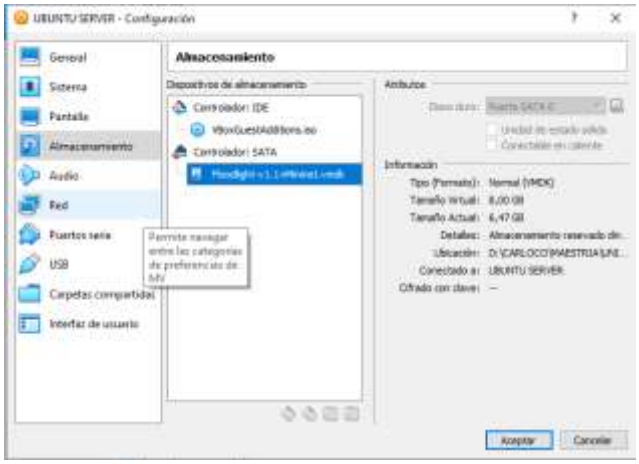


FIGURA 3.3. Importar máquina virtual Floodlight a Oracle VM VirtualBox.
FUENTE: Elaborado por el Autor.

3) Luego de importar la máquina virtual, es necesario habilitar dos tarjetas de red la primera en modo “Adaptador sólo-anfitrión” y otra en modo “Adaptador puente”, para poder ingresar mediante SSH (Secure Shell) a dicha máquina por la IP de dichas tarjetas y habilitar internet en la misma.

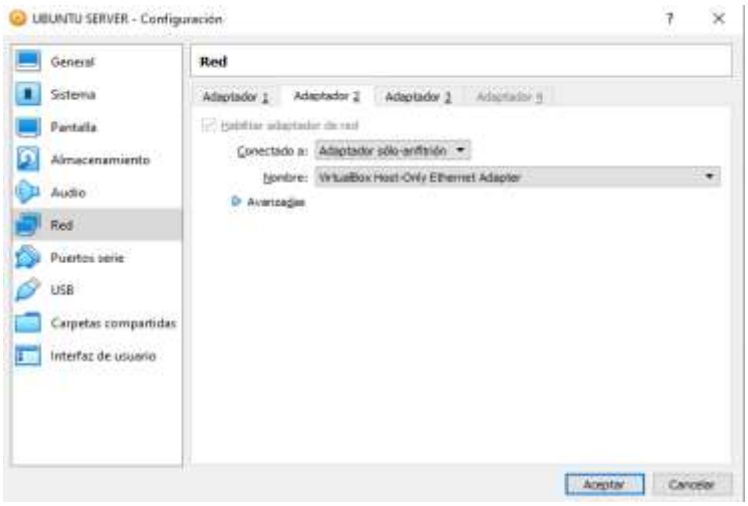


FIGURA 3.4. Integración tarjeta de red “Adaptador sólo-anfitrión”
FUENTE: Elaborado por el Autor.

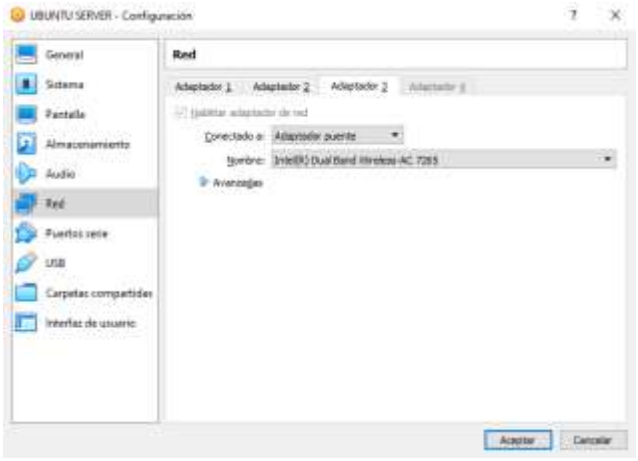


FIGURA 3.5. Integración tarjeta de red "Adaptador puente".
FUENTE: Elaborado por el Autor.

4) Finalmente se enciende la máquina virtual dando click en "Iniciar", y se ingresa a la misma con el user/password: Floodlight (Bonilla, 2016).

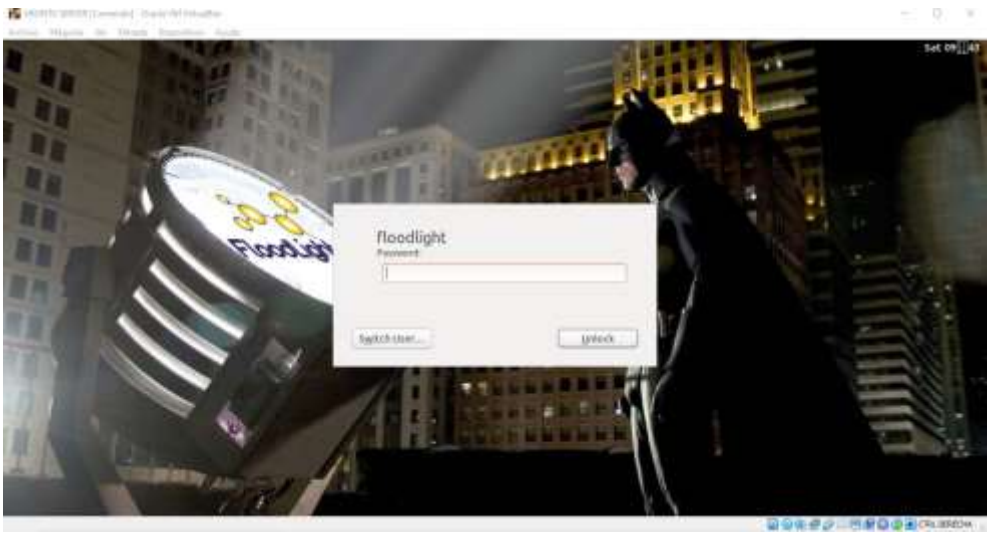


FIGURA 3.6. Encendido de VM Floodlight.
FUENTE: Elaborado por el Autor.

Para el ingreso mediante SSH a la máquina virtual se debe habilitar las tarjetas "Adaptador sólo-anfitrión" y "Adaptador puente" que se agregó en los pasos anteriores, para lo cual se ejecuta el siguiente comando dentro de la máquina (Bonilla, 2016).

```
floodlight@floodlight:~$ sudo dhclient eth1

floodlight@floodlight:~$ sudo dhclient eth2
```

Una vez ejecutado se comprueba si las tarjetas están habilitadas, con el comando:

```
floodlight@floodlight:~$ ifconfig
```

FIGURA 3.7. Tarjetas de red habilitadas.
FUENTE: Elaborado por el Autor.

3.2.4. Instalación de MININET

Para proceder con la instalación de MiniNet dentro de la máquina virtual Floodlight se debe realizar los siguientes pasos:

1) Se debe actualizar el sistema operativo que para el caso de estudio es UBUNTU por medio de los siguientes comandos.

```
floodlight@floodlight:~$ sudo apt-get update

floodlight@floodlight:~$ sudo apt-get upgrade

floodlight@floodlight:~$ sudo apt-get dist-upgrade
```

2) Una vez actualizado el sistema operativo es necesario instalar el software Git con el comando.

```
floodlight@floodlight:~$ sudo apt-get install git
```

3) Al tener instalado el software antes mencionado, se procede con la descarga del software MININET por medio del comando.

```
floodlight@floodlight:~$ git clone git://github.com/mininet/mininet
```

La cual crea una carpeta MININET en el directorio HOME con todos los archivos necesarios para su correcto funcionamiento.

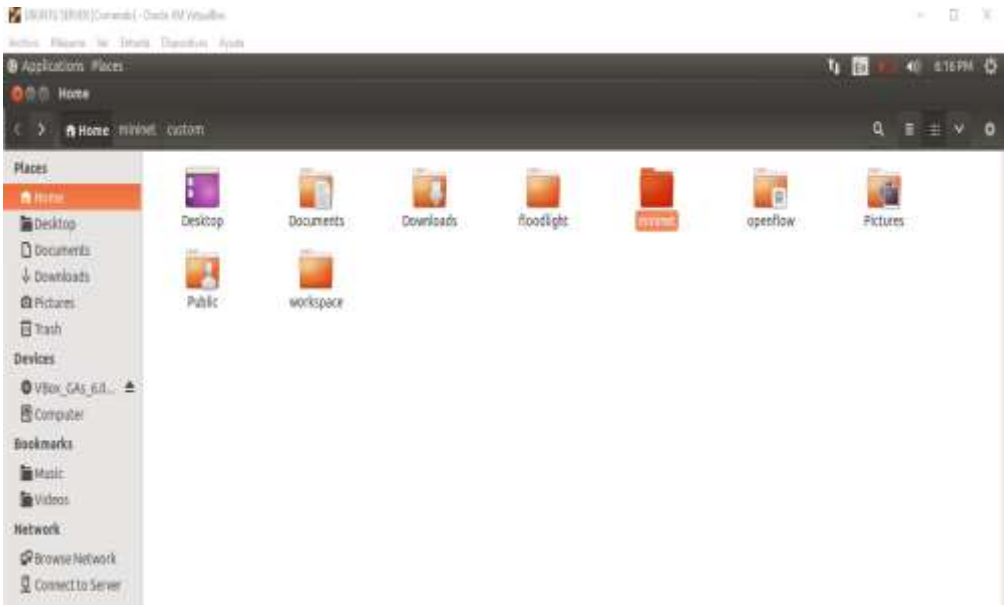


FIGURA 3.8. Instalación de MININET.
FUENTE: Elaborado por el Autor.

4) Para comprobar que la instalación fue completa se debe ingresar el siguiente comando.

```
floodlight@floodlight:~$ sudo mn --test pingall
```

El cual corre un pequeño escenario en MININET que debería dar un resultado exitoso

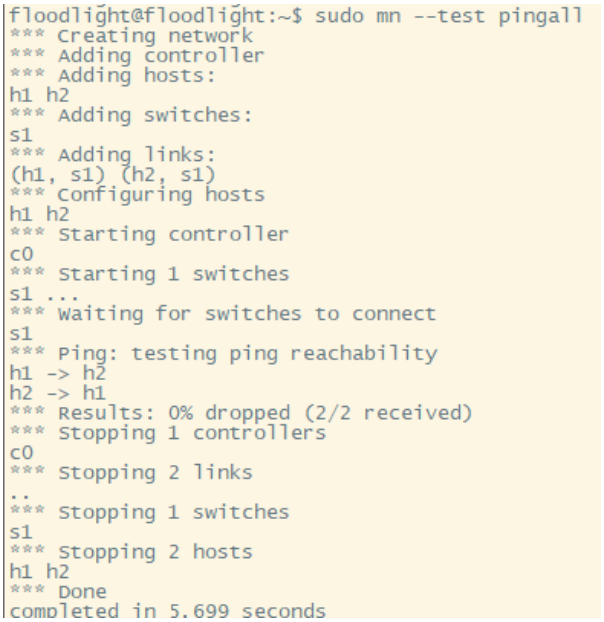


FIGURA 3.9. Comprobación de instalación de MININET.
FUENTE: Elaborado por el Autor.

3.2.5. Comandos de MININET

En la tabla 3.1 se muestran algunos comandos que se pueden ejecutar en la consola de MININET.

TABLA 3.1. Comandos de MININET

Comando	Descripción
net	Muestra información sobre la red.
h1 ping –c 1 h2	Manda un ping desde el Host 1 (h1) al Host 2 (h2).
pingall	Muestra la conectividad de la red.
h1 ifconfig -a	Muestra información sobre los interfaces de h1.
Xterm h1	Muestra un terminal de consola de Host 1 (h1).
dpctl dump-flow	Muestra el flujo que se produce en los switch.
exit	Cierra la consola.

FUENTE: (Pelicano, 2015) Modificado por el Autor.

3.2.6. Secure CRT

Para realizar la simulación de un ingreso remoto mediante el protocolo SSH se va a utilizar el aplicativo Secure CRT, el cual una vez instalado se debe configurar una sesión para que ingrese por medio de la IP de la tarjeta "Adaptador sólo-anfitrión", como se muestra en la figura 3.10 (Bonilla, 2016).

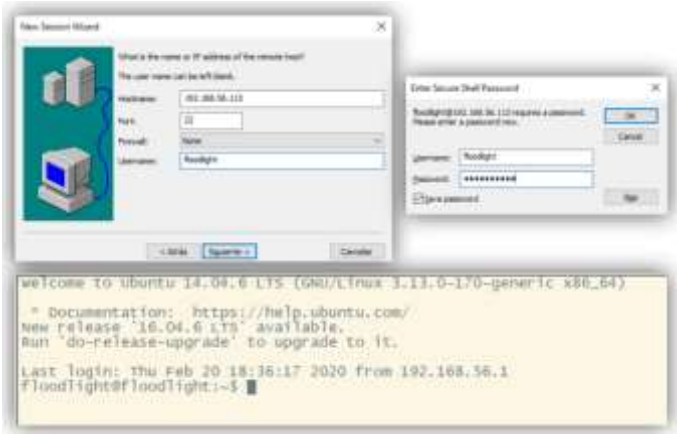


FIGURA 3.10. Configuración de conexión por protocolo SSH.
FUENTE: Elaborado por el Autor.

3.2.7. Topología de la simulación.

La topología de red que se utilizará para la simulación de redes SDN y probar la funcionalidad de control de acceso a nivel de capa 2 es la siguiente:

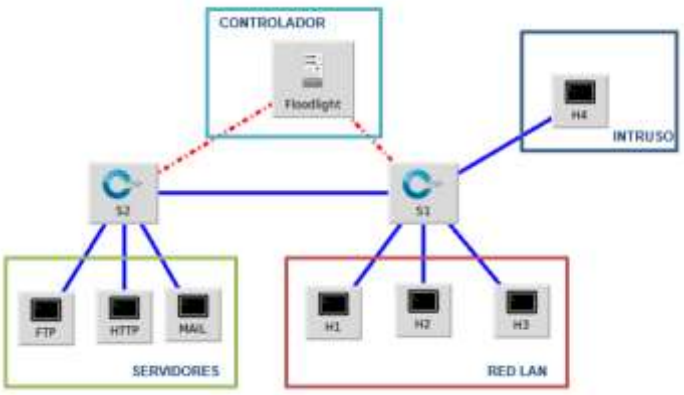


FIGURA 3.11. Topología de la Red SDN.
FUENTE: Elaborado por el Autor.

TABLA 3.2. Direccionamiento y MAC Address de la red SDN.

NOMBRE	IP	MAC
FTP	10.0.0.1	00:00:00:00:00:01
HTTP	10.0.0.2	00:00:00:00:00:02
MAIL	10.0.0.3	00:00:00:00:00:03
H1	10.0.0.4	00:00:00:00:00:04
H2	10.0.0.5	00:00:00:00:00:05
H3	10.0.0.6	00:00:00:00:00:06
H4	10.0.0.7	00:00:00:00:00:07

FUENTE: Elaborado por el Autor.

3.2.8. Configuración de la topología.

Para la configuración de la topología mostrada en el punto anterior será necesario el aplicativo MiniNet, el cual permite la creación de topologías personalizadas en lenguaje Python.

Este se ha desarrollado basado en la topología que viene por defecto cargada en MiniNet con nombre de archivo topo-2sw-2host.py.

El código que se muestra a continuación es la topología requerida, la misma que ha sido almacenada con el nombre de archivo topo-tesis.py (García & Rodríguez, 2014).

```
"""Custom topology Tesis
Adding the 'topos' dict with a key/value pair to generate our newly defined
topology enables one to pass in '--topo=mytopo' from the command line.
"""
from mininet.topo import Topo
class MyTopo( Topo ):
    "Topology Tesis."

    def build( self ):
        "Create custom topo."
        # Add hosts and switches
        H1 = self.addHost( 'h1' )
```

```
        H2 = self.addHost( 'h2' )
        H3 = self.addHost( 'h3' )
        H4 = self.addHost( 'h4' )
        HTTP = self.addHost( 'HTTP' )
        FTP = self.addHost( 'FTP' )
        MAIL = self.addHost( 'MAIL' )
        S1 = self.addSwitch( 's1' )
        S2 = self.addSwitch( 's2' )

        # Add links
            self.addLink( H1, S1 )
            self.addLink( H2, S1 )
            self.addLink( H3, S1 )
            self.addLink( H4, S1 )
            self.addLink( HTTP, S2 )
            self.addLink( FTP, S2 )
            self.addLink( MAIL, S2 )
            self.addLink( S1, S2 )


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

Para la ejecución de la topología por medio de MiniNet se utilizará el siguiente comando, dando como resultado lo mostrado en la figura 3.12.

floodlight@floodlight:~$ **sudo mn --custom topo-tesis.py --topo=mytopo --mac --switch ovsk --controller=remote,ip=10.0.2.15,port=6653 --link=tc**

```
*** Creating network
*** Adding controller
Unable to contact the remote controller at 10.0.2.15:6653
*** Adding hosts:
FTP HTTP MAIL h1 h2 h3 h4
*** Adding switches:
s1 s2
*** Adding links:
(FTP, s2) (HTTP, s2) (MAIL, s2) (h1, s1) (h2, s1) (h3, s1) (h4, s1) (s1, s2)
*** Configuring hosts
FTP HTTP MAIL h1 h2 h3 h4
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Starting CLI:
mininet>
```

FIGURA 3.12. Ejecución de la topología topo-tesis.py en Mininet.
FUENTE: Elaborado por el Autor.

3.2.9. Reglas de control de acceso a nivel de capa 2.

Basándose en la lista de direcciones MAC (Media Access Control) asignadas anteriormente se van a configurar las reglas de control de acceso a nivel de capa 2, con las cuales se permitirá el acceso entre los elementos de la red, ya que al habilitar el módulo toda la conectividad es bloqueada, por lo cual la meta está indicada en los siguientes puntos:

- HTTP, FTP y MAIL sin conectividad entre ellos.
- H1 y H2 tendrán acceso a los servidores y conectividad entre ellos.
- H3 solo tendrá acceso a los servidores HTTP y MAIL.
- H4 (Intruso) no tendrá acceso a ningún elemento de la red SDN.

Los comandos para habilitar y deshabilitar el módulo son los indicados a continuación.

```
curl http://10.0.2.15:8080/wm/firewall/module/enable/json -X PUT

curl http://10.0.2.15:8080/wm/firewall/module/disable/json -X PUT
```

Los comandos para crear las reglas y dar cumplimiento a la meta indicada son los mostrados a continuación (Pelicano, 2015).

```
curl -X POST -d '{"src-mac":"00:00:00:00:00:01", "dst-mac":"00:00:00:00:00:04"}'
http://10.0.2.15:8080/wm/firewall/rules/json
curl -X POST -d '{"src-mac":"00:00:00:00:00:04", "dst-mac":"00:00:00:00:00:01"}'
http://10.0.2.15:8080/wm/firewall/rules/json
curl -X POST -d '{"src-mac":"00:00:00:00:00:01", "dst-mac":"00:00:00:00:00:05"}'
http://10.0.2.15:8080/wm/firewall/rules/json
curl -X POST -d '{"src-mac":"00:00:00:00:00:05", "dst-mac":"00:00:00:00:00:01"}'
http://10.0.2.15:8080/wm/firewall/rules/json
curl -X POST -d '{"src-mac":"00:00:00:00:00:02", "dst-mac":"00:00:00:00:00:06"}'
http://10.0.2.15:8080/wm/firewall/rules/json
curl -X POST -d '{"src-mac":"00:00:00:00:00:06", "dst-mac":"00:00:00:00:00:02"}'
http://10.0.2.15:8080/wm/firewall/rules/json
```

```
curl -X POST -d '{"src-mac":"00:00:00:00:00:02", "dst-mac":"00:00:00:00:00:04"}'
http://10.0.2.15:8080/wm/firewall/rules/json
curl -X POST -d '{"src-mac":"00:00:00:00:00:04", "dst-mac":"00:00:00:00:00:02"}'
http://10.0.2.15:8080/wm/firewall/rules/json
curl -X POST -d '{"src-mac":"00:00:00:00:00:02", "dst-mac":"00:00:00:00:00:05"}'
http://10.0.2.15:8080/wm/firewall/rules/json
curl -X POST -d '{"src-mac":"00:00:00:00:00:05", "dst-mac":"00:00:00:00:00:02"}'
http://10.0.2.15:8080/wm/firewall/rules/json
curl -X POST -d '{"src-mac":"00:00:00:00:00:03", "dst-mac":"00:00:00:00:00:06"}'
http://10.0.2.15:8080/wm/firewall/rules/json
curl -X POST -d '{"src-mac":"00:00:00:00:00:06", "dst-mac":"00:00:00:00:00:03"}'
http://10.0.2.15:8080/wm/firewall/rules/json
curl -X POST -d '{"src-mac":"00:00:00:00:00:03", "dst-mac":"00:00:00:00:00:04"}'
http://10.0.2.15:8080/wm/firewall/rules/json
curl -X POST -d '{"src-mac":"00:00:00:00:00:04", "dst-mac":"00:00:00:00:00:03"}'
http://10.0.2.15:8080/wm/firewall/rules/json
curl -X POST -d '{"src-mac":"00:00:00:00:00:03", "dst-mac":"00:00:00:00:00:05"}'
http://10.0.2.15:8080/wm/firewall/rules/json
curl -X POST -d '{"src-mac":"00:00:00:00:00:05", "dst-mac":"00:00:00:00:00:03"}'
http://10.0.2.15:8080/wm/firewall/rules/json
curl -X POST -d '{"src-mac":"00:00:00:00:00:04", "dst-mac":"00:00:00:00:00:05"}'
http://10.0.2.15:8080/wm/firewall/rules/json
curl -X POST -d '{"src-mac":"00:00:00:00:00:05", "dst-mac":"00:00:00:00:00:04"}'
http://10.0.2.15:8080/wm/firewall/rules/json
```

Con el siguiente comando se podrá visualizar las reglas creadas dentro del módulo.

```
curl http://10.0.2.15:8080/wm/firewall/rules/json | python -mjson.tool
```

4. CAPÍTILO IV. SIMULACIÓN, PRUEBAS Y RESULTADOS.

Una vez establecidos los mecanismos necesarios, en este capítulo se va a realizar la simulación y las pruebas para verificar el cumplimiento de la meta indicada en el capítulo anterior.

4.1. Simulación de la red SDN.

Para iniciar la simulación de la red SDN se realizan los siguientes pasos:

A. Se debe abrir dos conexiones SSH hacia la máquina virtual por medio del programa Secure CRT.
B. En la primera sesión se va a ejecutar el controlador de la red SDN que en este caso de estudio es el controlador Floodlight aplicando los siguientes comandos.

```
floodlight@floodlight:~$ cd floodlight
floodlight@floodlight:~$ java -jar target/floodlight.jar
```

C. En la segunda sesión se ejecuta la simulación de la red SDN por medio de MiniNet con el siguiente comando.

```
floodlight@floodlight:~$ sudo mn --custom topo-tesis.py --topo=mytopo --mac --switch ovsk --controller=remote,ip=10.0.2.15,port=6653 --link=tc
```

4.2. Pruebas de la simulación de la red SDN.

Para poder diferenciar el comportamiento de la red SDN se realizarán dos pruebas: la primera sin control de acceso donde se espera que exista conectividad entre todos los equipos de la red y una segunda prueba donde se habilitará el control de acceso y se espera que la conectividad se pierda entre algunos equipos de la red cumpliendo la meta planteada en el capítulo anterior.

4.2.1. Prueba sin control de acceso.

En esta prueba se simulará la red SDN sin habilitar las políticas de control de acceso con lo cual se desea comprobar la conectividad entre todos los elementos de la red SDN.

```
floodlight@floodlight:~$ cd floodlight
floodlight@floodlight:~/floodlight$ java -jar target/floodlight.jar
22:09:16.201 INFO [n.f.c.m.FloodlightModuleLoader:main] Loading modules from src/main/resources/floodlightdefault.properties
22:09:16.394 WARN [n.f.r.RestApiServer:main] HTTPS disabled; HTTPS will not be used to connect to the REST API.
22:09:16.395 WARN [n.f.r.RestApiServer:main] HTTP enabled; Allowing unsecure access to REST API on port 8080.
22:09:17.610 WARN [n.f.c.i.OFSwitchManager:main] SSL disabled. Using unsecure connections between Floodlight and switches.
22:09:17.610 INFO [n.f.c.i.OFSwitchManager:main] Clear switch flow tables on initial handshake as master: TRUE
22:09:17.611 INFO [n.f.c.i.OFSwitchManager:main] Clear switch flow tables on each transition to master: TRUE
22:09:17.611 INFO [n.f.c.i.OFSwitchManager:main] Setting 0x4 as the default max table to receive table-miss flow
22:09:17.625 INFO [n.f.c.i.OFSwitchManager:main] Setting max table to receive table-miss flow to 0x4 for DPID 00:00:00:00:00:00:00:01
22:09:17.626 INFO [n.f.c.i.OFSwitchManager:main] Setting max table to receive table-miss flow to 0x4 for DPID 00:00:00:00:00:00:00:02
22:09:17.630 INFO [n.f.c.i.Controller:main] Controller role set to ACTIVE
22:09:17.672 INFO [n.f.f.Forwarding:main] Default hard timeout not configured. Using 0.
22:09:17.672 INFO [n.f.f.Forwarding:main] Default idle timeout not configured. Using 5.
22:09:17.673 INFO [n.f.f.Forwarding:main] Default priority not configured. Using 1.
22:09:17.673 INFO [n.f.f.Forwarding:main] Default flags will be empty.
22:09:17.674 INFO [n.f.f.Forwarding:main] Default flow matches set to: VLAN=true, MAC=true, IP=true, TPPT=true
22:09:18.176 INFO [o.s.s.i.c.FallbackCCProvider:main] Cluster not yet configured; using fallback local configuration
```

FIGURA 4.1. Ejecución de controlador Floodlight.
FUENTE: Elaborado por el Autor.

Por medio del comando ping en la consola de MiniNet se verificará si existe conectividad entre los elementos de la red SDN, como se muestra en la figura 4.2 (Bonilla, 2016).

```
mininet> HTTP ping -c 5 h4
PING 10.0.0.7 (10.0.0.7) 56(84) bytes of data.
64 bytes from 10.0.0.7: icmp_seq=1 ttl=64 time=13.6 ms
64 bytes from 10.0.0.7: icmp_seq=2 ttl=64 time=0.220 ms
64 bytes from 10.0.0.7: icmp_seq=3 ttl=64 time=0.141 ms
64 bytes from 10.0.0.7: icmp_seq=4 ttl=64 time=0.037 ms
64 bytes from 10.0.0.7: icmp_seq=5 ttl=64 time=0.047 ms

--- 10.0.0.7 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 0.037/2.818/13.647/5.414 ms
```

FIGURA 4.2. Prueba de ping entre dos elementos de red SDN
FUENTE: Elaborado por el Autor.

Para realizar una comprobación más rápida de la conectividad de los hosts de la red se utilizará el comando pingall, cuyo resultado es el mostrado en la figura 4.3, en la cual se puede observar que existe conectividad entre todos los elementos de la red SDN.

```
mininet> pingall
*** Ping: testing ping reachability
FTP -> HTTP MAIL h1 h2 h3 h4
HTTP -> FTP MAIL h1 h2 h3 h4
MAIL -> FTP HTTP h1 h2 h3 h4
h1 -> FTP HTTP MAIL h2 h3 h4
h2 -> FTP HTTP MAIL h1 h3 h4
h3 -> FTP HTTP MAIL h1 h2 h4
h4 -> FTP HTTP MAIL h1 h2 h3
*** Results: 0% dropped (42/42 received)
```

FIGURA 4.3. Prueba de ping de todos los elementos de red SDN
FUENTE: Elaborado por el Autor.

En la figura 4.4 se puede apreciar la topología de la red SDN de forma gráfica aprovechando una de las ventajas que da la controladora Floodlight.

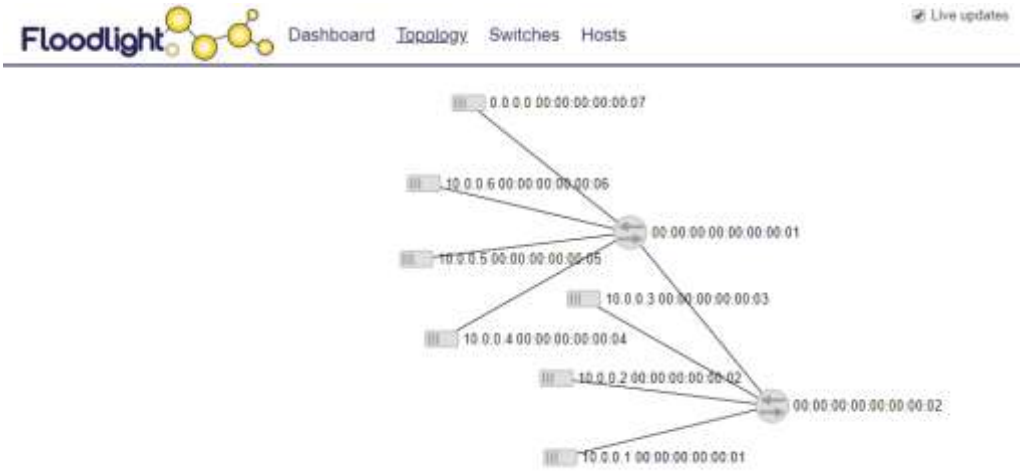


FIGURA 4.4. Topología de la red SDN en interfaz web.
FUENTE: Elaborado por el Autor.

4.2.2. Prueba con control de acceso.

Esta prueba se realizará ejecutando las políticas de control de acceso, una vez que se ha comprobado que si existe conectividad entre todos los elementos de la red SDN en el punto anterior.

Para esto se requiere abrir una tercera conexión SSH a la máquina virtual donde se ingresará el comando que habilitará el módulo de control de acceso mostrado en la figura 4.5.

```
floodlight@floodlight:~$ curl http://10.0.2.15:8080/wm/firewall/module/enable/json -X PUT
{"status" : "success", "details" : "firewall running"}floodlight@floodlight:~$
```

FIGURA 4.5. Comando para habilitar módulo de control de acceso.
FUENTE: Elaborado por el Autor.

Si se habilita el módulo de control de acceso sin añadir reglas da como resultado que no exista conectividad en la red SDN lo que se verifica por medio del comando pingall, como se observa en la figura 4.6.

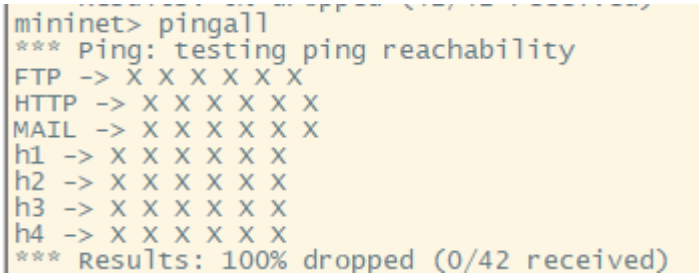


FIGURA 4.6. Prueba de conectividad solo habilitado el módulo de control de acceso
FUENTE: Elaborado por el Autor.

Después se ejecutarán las reglas de control de acceso mostradas en el capítulo anterior y por medio de los comandos ping y pingall se verificará que la conectividad se cumpla según lo indicado, esto se muestra en las figuras 4.7 y 4.8 respectivamente.

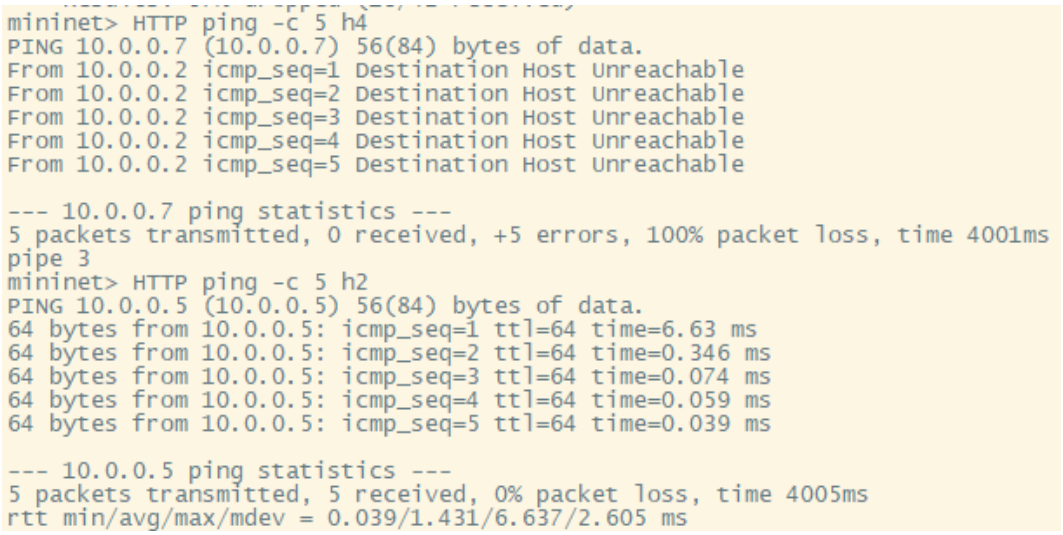


FIGURA 4.7. Verificación de ping con control de acceso habilitado.
FUENTE: Elaborado por el Autor.

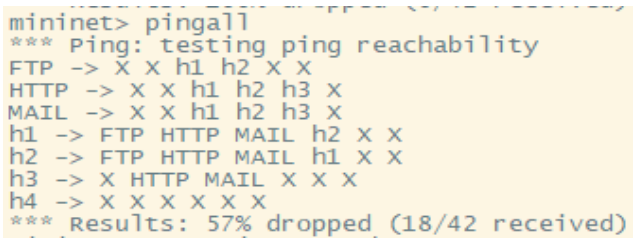


FIGURA 4.8. Prueba de ping de todos los elementos habilitado control acceso.
FUENTE: Elaborado por el Autor.

4.3. Análisis de resultados.

En este punto se realizará el análisis de las pruebas ejecutadas en el punto anterior.

4.3.1. Análisis de la prueba sin control de acceso.

En la primera prueba se puede comprobar el correcto funcionamiento del controlador Floodlight, ya que este permite la conectividad de todos los elementos de la red SDN y mediante el comando **dpctl** se puede visualizar el flujo de los switch de la red SDN como se muestra en la figura 4.9.

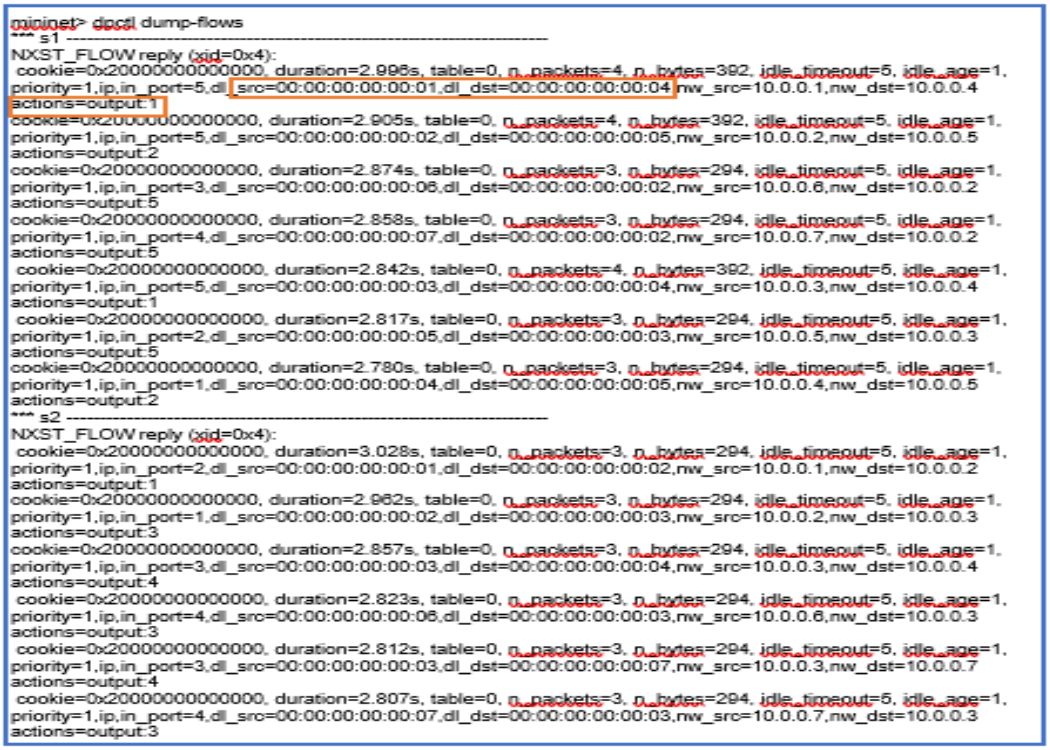


FIGURA 4.9.Flujo de paquetes en switch OpenFlow.
FUENTE: Elaborado por el Autor.

En esta figura se puede observar que tanto en el switch 1 como el 2 no existe un bloqueo en la transmisión de paquetes entre los dispositivos de la red SDN.

4.3.2. Análisis de la prueba con control de acceso.

En esta prueba el comportamiento del módulo de control de acceso es el esperado, ya que cumple con la meta planteada en el capítulo anterior, esto

quiere decir, que los equipos que están indicados en las reglas tienen conectividad entre ellos, caso contrario el paquete es dropeado y por medio del comando **dpctl** se visualizará el flujo de los switch de la red SDN como se muestra en la figura 4.10.

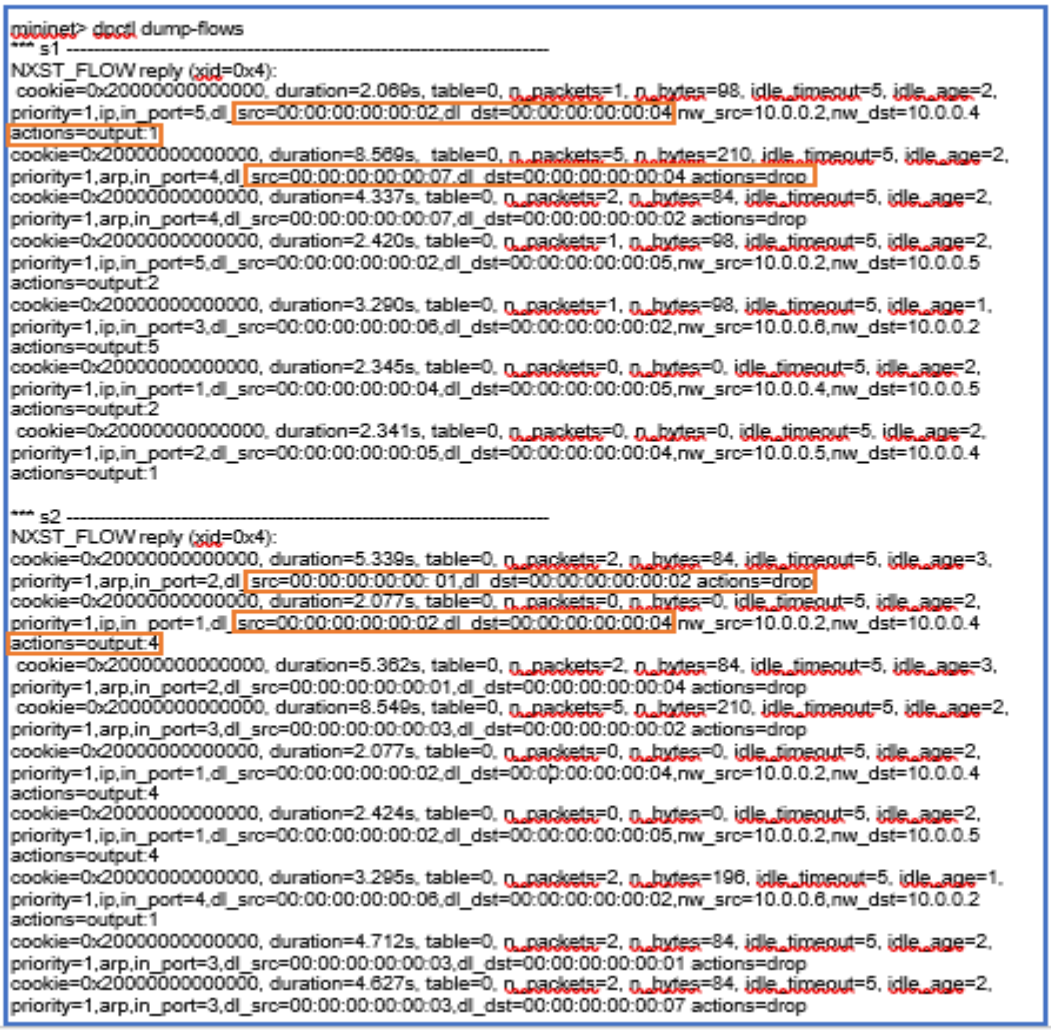


FIGURA 4.10. Flujo de paquetes en switch OpenFlow con módulo de control de acceso.
FUENTE: Elaborado por el Autor.

CONCLUSIONES

- La investigación realizada sobre la tecnología SDN facilitó la comprensión y el proceso de simulación de este tipo de redes, adicionalmente cabe indicar que las redes SDN son escalables, gestionables y seguras.
- Mediante la caracterización de los controladores SDN se pudo conocer algunos de ellos, y basándose en las características investigadas se seleccionó el controlador Floodlight para la simulación de red SDN por su fácil configuración, alto rendimiento y porque viene con módulos cargados, facilitando su uso.
- El emulador MiniNet es una herramienta muy útil para la simulación de redes SDN, que permite obtener resultados muy cercanos a la realidad, adicionalmente su compresión de funcionalidad es sencilla y posee comandos útiles para la verificación de su operatividad.
- Al habilitar el módulo de control de acceso a nivel de capa 2 mediante la utilización del controlador Floodlight, se bloquea toda la red SDN, lo que garantiza la seguridad y facilita la configuración del plano de control de los dispositivos.
- Gracias a la combinación del controlador Floodlight y el emulador MiniNet se pudo comprobar la funcionalidad de control de acceso a nivel de capa 2, demostrando que se puede tener un método de seguridad en las redes SDN, cumpliendo de esta manera la hipótesis planteada en esta investigación.

RECOMENDACIONES

- Utilización del emulador MiniNet, ya que, por su fácil instalación, es una herramienta práctica para experimentar con redes SDN, para de esta manera familiarizarse con los conceptos de SDN y OpenFlow, para que en un futuro se pueda realizar otros estudios con mayor complejidad.
- Realizar pruebas con el controlador Floodlight en el cual vienen incorporados módulos de control de red SDN, como por ejemplo ver la factibilidad de dar seguridad a la red SDN a nivel de capa 3.
- Seguir realizando estudios sobre las redes SDN para seguir desarrollando más información sobre las funcionalidades que esta tecnología puede brindar, para de esta manera, dar otra solución a las redes LAN, como en este caso de estudio que presenta una solución alternativa en la seguridad de las redes LAN que podría pertenecer a pequeñas y medianas empresas donde el número de elementos de la red no sea numeroso.

GLOSARIO

API

Application Programming Interface

CLI

Command Line Interface

DNS

Domain Name System

LAN

Local Area Network

MAC

Media Access Control

ONF

Open Networking Foundation

OXM

OpenFlow Extensible Match

SDN

Software-Defined Networking

SSH

Secure Shell

SSL

Secure Sockets Layer

TLV

Type Lenght Value

VLAN

Virtual Local Area Network

BIBLIOGRAFÍA

Chang, C.-H., & Lin, Y.-D. (2015). *OpenFlow Version Roadmap*. Retrieved from speed.cis.nctu.edu.tw: http://speed.cis.nctu.edu.tw/~ydlin/miscpub/indep_frank.pdf

electiva redes avanzadas. (2015). *OPENFLOW*. Retrieved from electivaredesavanzadas.blogspot.com: http://electivaredesavanzadas.blogspot.com/2015/02/funcionamiento-de-openflow-openflow.html

IRIS. (2020). *OpenIRIS*. Retrieved from openiris.etri: https://openiris.etri.re.kr/

Izard, R. (2015). *Firewall REST API*. Retrieved from Floodlight Controller: https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343614/Firewall+REST+API

Linkletter, B. (2014). *How to install the Mininet SDN Network Simulator*. Retrieved from Open-Source Routing and Network Simulation: http://www.brianlinkletter.com/how-to-install-mininet-sdn-network-simulator/

Marín, Y. (2016). *Estructura y componentesde un conmutador OpenFlow*. Retrieved from researchgate: https://www.researchgate.net/figure/Figura-18-Estructura-y-componentes-de-un-conmutador-OpenFlow-Los-conmutadores-OpenFlow_fig1_312187600

Millán, R. (2014). *SDN: el futuro de las redes inteligentes.* Retrieved from Ramon Millan: https://www.ramonmillan.com/tutoriales/sdnredesinteligentes.php

Mininet Team. (2018). *Descripción general de Mininet*. Retrieved from mininet.org: http://mininet.org/overview/

Project Floodlight. (2020). *Floodlight*. Retrieved from projectfloodlight: http://www.projectfloodlight.org/floodlight/

Puntinformatic. (s.f.). *Beneficios de la red definida por software (SDN)*. Retrieved from puntinformatic: https://puntinformatic.com/beneficios-de-la-red-definida-por-software-sdn/

Ríos, R. A. (2016). *Conceptualización de SDN y NFV*. Retrieved from scielo: http://scielo.senescyt.gob.ec/scielo.php?script=sci_arttext&pid=S1390-67122016000100029#B8

SICROM. (s.f.). *Las características especiales de las SDN*. Retrieved from SICROM: http://sicrom.com/blog/las-caracteristicas-especiales-de-las-sdn/

Sulca, M. I. (2018). *Redes definidas por Software (SDN).* Retrieved from UNIVERSIDAD COMPLUTENSE MADRID: https://informatica.ucm.es/data/cont/media/www/pag-103596/transparencias/redes-por-software-SDN.pdf

Wang, Q. (2018). *Floodlight VM*. Retrieved from floodlight.atlassian.net: https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/8650780/Floodlight+VM#FloodlightVM-Download

DECLARACIÓN Y AUTORIZACIÓN

Yo, **Carlos Alberto Carrillo Rodas**, con C.C: # **0603321399** autor del trabajo de Maestría titulada: **Simulación de una red definida por software (SDN) para el control de acceso de los elementos de la red a nivel de capa 2,** previo a la obtención del título de **Magíster en Telecomunicaciones** en la Universidad Católica de Santiago de Guayaquil.

1.- Declaro tener pleno conocimiento de la obligación que tienen las instituciones de educación superior, de conformidad con el Artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de titulación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la SENESCYT a tener una copia del referido trabajo de titulación, con el propósito de generar un repositorio que democratice la información, respetando las políticas de propiedad intelectual vigentes.

Guayaquil, a los 29 días del mes de junio de 2020

f. _____

Nombre: **Carlos Alberto Carrillo Rodas**

C.C: 0603321399

REPOSITORIO NACIONAL EN CIENCIA Y TECNOLOGÍA		
FICHA DE REGISTRO DE TESIS/TRABAJO DE TITULACIÓN		
TÍTULO Y SUBTÍTULO:	Simulación de una red definida por software (SDN) para el control de acceso de los elementos de la red a nivel de capa 2	
AUTOR(ES)	Carlos Alberto Carrillo Rodas	
REVISOR(ES)/TUTOR	MSc. Edgar Quezada Calle; MSc. Luis Córdova Rivadeneira / MSc. Manuel Romero Paz	
INSTITUCIÓN:	Universidad Católica Santiago de Guayaquil	
FACULTAD:	Sistema de Posgrado	
PROGRAMA:	Maestría en Telecomunicaciones	
TITULO OBTENIDO:	Magister en Telecomunicaciones	
FECHA DE PUBLICACIÓN:	Guayaquil, 29 de junio de 2020	No. DE PÁGINAS: 56
ÁREAS TEMÁTICAS:	Redes SDN, Protocolo OpenFlow, Control de Acceso, Seguridad de la red, Secure CRT,	
PALABRAS CLAVES/ KEYWORDS:	SDN, OpenFlow, MiniNet, Floodlight, POX, OpenIRIS, Oracle VM VirtualBox	

RESUMEN/ABSTRACT:

El siguiente caso de estudio se divide en cuatro capítulos: en el primero se describe la investigación a realizar como son los objetivos planteados, el planteamiento del problema y la hipótesis a analizar, en el siguiente capítulo se realiza el estudio del arte de la redes SDN, protocolo OpenFlow, así como de los tipos de controladores SDN, parte teórica fundamental para el diseño de la red SDN la cual está definida en el capítulo tres, así como todas las herramientas necesarias de hardware y de software para comprobar su funcionamiento, como último capítulo se tiene la simulación, pruebas y análisis de la misma en una red definida por software para el control de acceso de los elementos de la red a nivel de capa 2. El tipo de investigación a desarrollar es exploratoria, por la recolección de conocimientos previos sobre redes SDN, descriptiva porque se va caracterizar el objeto de estudio, aplicada ya que se utilizarán los conocimientos obtenidos en la investigación para la simulación de una red SDN para el control de acceso de los elementos de la red a nivel de capa 2, y transversal ya que los resultados se los realizará en un momento y tiempo definido. La metodología a aplicarse es el método deductivo - inductivo para la recopilación y caracterización de la información, analítico - sintético para descomponer el objetivo principal en varios objetivos específicos, para al finalizar recopilar la información y por medio de la síntesis encontrar la solución al problema, y con un enfoque cuantitativo con el cual mediante pruebas se desea obtener información de la funcionalidad de una de sus característica.

ADJUNTO PDF:	☒ SI	☐ NO
CONTACTO CON AUTOR/ES:	**Teléfono:** +593-995307972	**E-mail:** carlangas_sp@hotmail.com
	Nombre: Romero Paz Manuel de Jesús	
	Teléfono: +593-994606932	

CONTACTO CON LA INSTITUCIÓN (COORDINADOR DEL PROCESO UTE):	**E-mail:** manuel.romero@cu.ucsg.edu.ec
SECCIÓN PARA USO DE BIBLIOTECA	
Nº. DE REGISTRO (en base a datos):	
Nº. DE CLASIFICACIÓN:	
DIRECCIÓN URL (tesis en la web):	