RESUMEN – TRABAJO DE GRADO

AUTORES	**CRISTIAN ANDRES AFANADOR DELGADO**
FACULTAD	**INGENIERIAS**
PLAN DE ESTUDIOS	**INGENIERIA DE SISTEMAS**
DIRECTOR	**FABIAN RANULFO CUESTA QUINTERO**
TÍTULO DE LA TESIS	**IMPLEMENTACIÓN DE UN CONTROLADOR EN SOFTWARE LIBRE PARA REDES DEFINIDAS POR SOFTWARE MEDIANTE LA HERRAMIENTA MININET, QUE PERMITA REALIZAR SIMULACIONES CON PROTOCOLO IPv6**

RESUMEN
(70 palabras aproximadamente)

ESTE PROYECTO DE INVESTIGACIÓN TIENE COMO OBJETIVO REALIZAR UNA REVISIÓN SOBRE LOS DIFERENTES CONTROLADORES LIBRES Y CUÁLES DE ELLOS SOPORTAN EL PROTOCOLO IPV6, PARA DE ESTA MANERA EMULAR UN AMBIENTE VIRTUAL CONTROLADOR POR MEDIO DEL SOFTWARE MININET E IMPLEMENTARLO EN EL LABORATORIO DE REDES Y TELECOMUNICACIONES DE LA UNIVERSIDAD FRANCISCO DE PAULA SANTANDER OCAÑA, CON LA FINALIDAD DE ANALIZAR SU COMPORTAMIENTO FRENTE A UNA RED TRADICIONAL Y LA POSIBLE DE MIGRACIÓN A ESTA TECNOLOGÍA.

CARACTERÍSTICAS

PÁGINAS: 93	PLANOS:	ILUSTRACIONES: 45	CD-ROM: 1

IMPLEMENTACIÓN DE UN CONTROLADOR EN SOFTWARE LIBRE PARA REDES DEFINIDAS POR SOFTWARE MEDIANTE LA HERRAMIENTA MININET QUE PERMITA REALIZAR SIMULACIONES CON PROTOCOLO IPv6

Autor:

CRISTIAN ANDRES AFANADOR DELGADO

Trabajo de investigación para optar al título de Ingeniero de Sistemas

Director:

FABIAN ARNULFO CUESTA QUINTERO

Ingeniero de Sistemas

UNIVERSIDAD FRANCISCO DE PAULA SANTANDER OCAÑA

FACULTAD DE INGENIERÍAS

INGENIERÍA DE SISTEMAS

Ocaña, Colombia **agosto de 2018**

AGRADECIMIENTOS

Agradezco principalmente a Dios, por guiarme en mi camino y ser el motor espiritual para continuar la realización de este proyecto, por no dejarme desfallecer en momentos y darme ánimos cuando las cosas no salían como se esperaban.

A nuestra preciada Universidad Francisco de Paula Santander Ocaña a la cual llevo presente, por brindarme la oportunidad de estar en sus aulas de conocimiento y formarme como un profesional.

De manera especial a mi director Ms. Fabián Cuesta Quintero por brindarme su confianza y la posibilidad de trabajar y realizar este proyecto. A los profesores que han dedicado parte de su tiempo en brindarme los conocimientos necesarios para llegar a esta instancia de mi carrera y a esos compañeros de estudio que me apoyaron y se esforzaron conmigo en esta formación profesional.

A mis padres, que son el pilar fundamental de lo que soy, tanto formativamente y como persona además de su apoyo incondicional en todo momento.

Todo este trabajo ha sido posible gracias a ellos.

Índice

Lista de Tablas

Lista de Figuras

Resumen

Este proyecto de investigación tiene como objetivo realizar una revisión sobre los diferentes controladores libres y cuáles de ellos soportan el protocolo IPv6, para de esta manera emular un ambiente virtual controlador por medio del software Mininet e implementarlo en el laboratorio de redes y telecomunicaciones de la Universidad Francisco de Paula Santander Ocaña, con la finalidad de analizar su comportamiento frente a una red tradicional y la posible de migración aesta tecnología.

Palabras claves

Openflow, Red definida por software, POX, Beacon, Virtualización de red, Controlador SDN, Switch, Router, Servidor, Virtual switch, Mininet.

Introducción

Debido al crecimiento de las redes en la comunicación se ha buscado maneras para facilitar la configuración de los dispositivos en una red y a su vez dar seguridad a la información que transita por ella, muchas de empresas de gran índole como lo son CISCO, HP, HUAWEI entre otras han apostado a las redes definidas por software como una solución a este acontecimiento incursionando en este campo.

Por tal motivo, este proyecto tiene como propósito emular una red definida por software mediante Mininet junto con el protocolo OpenFlow bajo soporte el protocolo IPv6, para un análisis comparativo con respecto a la red tradicional, llevándose a cabo con un servidor hewlett packard teniendo como sistema operativo Ubuntu 17.0, identificando los distintos tiempos te conexión que se lleva de un host a otro.

En el presente documento se mostrará diferentes tipos de controladores de carácter libre que pueden ser utilizados para estos fines y una comparativa entre los mismos, además, se expondrá desde la instalación, integración y aplicación de los elementos necesarios para la implementación y se dará a conocer los análisis y sugerencias a los que conllevan esta investigación.

Capítulo 1. Implementación de un controlador en software libre para redes definidas por software mediante la herramienta Mininet, que permita realizar simulaciones con el protocolo IPv6

1.1 Planteamiento del problema

Las redes IP convencionales han dado solución por largos años a la conectividad de los usuarios. Para realizar esta acción, se debe pasar por distintos dispositivos como lo son: los switch, router, firewall, entre otros. Cada uno de estos son independiente, tomando decisiones individuales por cada paquete de red que lleguen a ellos, haciendo así que su configuración y/o mantenimiento sea más difícil debido a que no se encuentra centralizado. En la revolución de las tecnologías de la información (TI) se encuentran inmersas la virtualización de servidores y almacenamiento, pero se ha encontrado una limitante, la cual es la estructura de red tradicional, que no permite conexiones ágiles con la nube, a su vez este mismo tipo de redes evoluciona lentamente y tienen limitaciones con respecto a sus funcionalidades en cuanto a proveedores como la ASIC y los proveedores de los dispositivos de red, por su naturaleza estática, generando un alto nivel en gastos de operación (OPEX) (Citrix Systems, 2014).

La necesidad que se genera, es estudiada por grandes empresas como lo son, CISCO afirma, que, en la mayoría de las grandes redes empresariales, los dispositivos de red abarcan las funciones de datos y de control, haciendo más difícil el ajuste de la infraestructura y el

funcionamiento de la red a la adición a gran escala de sistemas finales, máquinas virtuales y redes virtuales. (CISCO, 2013), además, Hewlett Packard menciona que se puede liberar a los administradores de TI de la monotonía de la configuración de red que se hace de forma manual y su reconfiguración ya que la red se puede ajustar automáticamente a las necesidades de los negocios y así el personal puede enfocarse más en la calidad del negocio experiencia, y dedicar menos tiempo a administrar los detalles de la infraestructura de red (Hewlett-Packard Development Company, L.P, 2013) , por su lado Huawei destaca tres dificultades que presenta las redes tradicionales, falta de garantías de experiencia del usuario, implementación ineficiente de servicios, y lenta adaptación a nuevos servicios (Huawei Technologies Co, 2018), además estos aseguran que las rede definidas por software son una gran alternativa de implementación, además, su enfoque programable para diseñar, elaborar y administrar redes de datos, dando administración, separando el plano de datos y el plano de control, obteniendo una visión general de la red, proporcionando servicios de Qos, enrutamiento, control de acceso, almacenamiento entre otras. (Big Switch Networks,2014), nos presentan grandes beneficios a la hora de instalar una red.

Este proyecto está orientado a la comparación, selección e implementación de un controlador SDN en el laboratorio de redes y telecomunicaciones de la Universidad Francisco de Paula Santander Ocaña, con el propósito de analizar el comportamiento de la red OpenFlow frente una red tradicional. Debido a la ausencia de equipos que soportan el protocolo, se opta por una implementación virtual, emulada en un servidor con plataforma libre como lo es Linux Ubuntu.

1.2 Formulación del problema

¿Se podrá implementar un controlador de redes definidas por software de forma libre que permita el manejo del protocolo IPV6?

1.3 Objetivos de investigación

1.3.1 Objetivo general.

Implementar un controlador en software libre para redes definidas por software mediante la herramienta Mininet que permita realizar simulaciones con el protocolo IPv6.

1.3.2 Objetivos específicos.

Realizar un comparativo entre los controladores de software libre, para la selección del que mejor se adapta al protocolo IPv6.

Desarrollar un ambiente controlado mediante la herramienta Mininet que permita validar el funcionamiento del controlador aplicado al protocolo IPv6, tomando como referencia las necesidades del laboratorio de redes y telecomunicaciones de la Universidad Francisco de Paula Santander Ocaña.

Analizar los resultados obtenidos en el caso de estudio planteado, para la identificación de las ventajas y desventajas de las redes definidas por software bajo el protocolo IPv6.

1.4 Justificación

Para la Universidad es de vital importancia poder avanzar y estar actualizada en la última tecnología y a su vez contribuir al mejoramiento del ambiente. Hoy en día las redes reciben cada vez más atención por lo tanto estas requieren mayor velocidad en el envío de su información y por ello surgen las Software Network Defined (SDN), el cual, al ser programable, separa el plano de datos y el de control, da mayor fluidez en la entrega de sus paquetes generando que la aplicación de las mismas dará mayor sencillez en la arquitectura de red que una convencional (Popoviciu, 2013). En la actualidad se considera que las redes definidas por software (SDN) como una evolución fundamental de la tecnología, a su vez contrarresta el aumento de la complejidad y costos de gestión y funcionamiento de las redes tradicionales, y da introducción a nuevos servicios o tecnologías.

La virtualización de las redes permite adecuar las necesidades de la red a nuevas situaciones, con ello las redes se benefician, ya que se puede aplicar donde y cuando se necesiten sin la necesidad de esperar a complejos despliegues manuales, ya sean en configuración, compatibilidades entre otras. Es importante destacar que esta tecnología está siendo estudiada por grandes empresas como lo son CISCO, HP, HUAWEI, lo cual permitirá a la Universidad Francisco de Paula Santander Ocaña, tener un tema de índole actual donde las mismas entidades aseguran que esta tecnología será de muy buena implementación.

En la parte operativa, su aplicación es más sencilla que una red tradicional, ya que al estar centralizado su configuración en el controlador y en las aplicaciones, hace que la estructura de

red sea más eficiente y gracias a esto los costos de mantenimiento serán reducidos ya que solo se centran en el controlador.

1.5 Hipótesis

Si se implementan el protocolo ipv6 en las redes definidas por software, se podrá observar el mejoramiento del rendimiento en las conexiones de red.

Al implementar las redes definidas por software, ya no se deberán hacer uno a uno las configuraciones en la infraestructura de red.

SDN simplifica el tiempo de implementación de cambios sobre la infraestructura de red.

Al aplicar SDN se puede generar una mayor seguridad a la infraestructura de red.

1.6 Limitaciones

1.6.1 Delimitación operativa. En primera instancia, se llevará a cabo la implementación del controlador Openflow sobre un ordenador con sistema operativo Ubuntu 16.0.4, instalando en él un software de emulación de redes (Mininet), en el cual, se utilizará el protocolo IPV6 en las redes definidas por software (SDN), por medio de los controladores Seleccionados en el caso de estudio propuesto. Así mismo.

1.6.2 Delimitación conceptual. Para la realización del proyecto, se tendrán en cuenta el siguiente vocabulario:

Openflow, Red definida por software, POX, Beacon, Virtualización de red, Controlador SDN, Switch,Router, Servidor, Virtual switch, Mininet.

1.6.3 Delimitación geográfica. El proyecto se desarrolló en el laboratorio de redes y telecomunicaciones de la Universidad Francisco de Paula Santander Ocaña, implementado en un ambiente virtual a través de Mininet.

1.6.4 Delimitación temporal. El proyecto se llevará a cabo en un periodo de tiempo de 6 meses, contados a partir de su aprobación.

Capítulo 2. Marco referencial

2.1 Marco histórico

La evolución de la tecnología va cada vez más rápida y consigo trae nuevos paradigmas que deben ser aplicados según las necesidades y la adaptabilidad de las personas, un ejemplo claro de ellos es la migración del protocolo IPv4 a IPv6, ya que es necesaria, pero todavía hay resistencia de muchos usuarios a dicha migración. Debido a los cambios que han tenido el ambiente tecnológico y la integración de la misma con la nube, las redes especialmente han afrontado nuevos desafíos, uno de ellos es en su estructura, la cual al asociarse con los cambios anteriormente mencionados han decidido apostarle a la aplicación de las Redes Definidas por Software SDN (software Definition Network).

La idea de hacer redes programables no es esencialmente de las redes definidas por software ya que se encuentran antecedentes donde se había propuesto, como lo fue SOFTNET la cual en los años 80 propuso una red multisalto semejantes a las actuales (Wireless Sensor Networks), esta se basaba en que cada campo de dato de cada paquete incluían comandos, las cuales los nodos iban ejecutando a medida que los iban recibiendo, esto generaba las que la red fuera dinámica y se modifica en tiempo real, esto fue un intento por crear una red auto-organizarle, por lo cual generó la experimentación e innovación en distintos protocolos. Aunque este proyecto no se siguió produciendo fue la base para la posterior evolución Active Networks.

Las Active networks presentaban una arquitectura consistente los cuales en los paquetes llevaban pequeños programas que podían ser ejecutadas por los nodos por donde pasaban por lo

tanto los switch y los routers podían procesar los paquetes de datos, por lo tanto, podían participar en los mensajes.

Dada las relaciones de SOFTNET y las Active Networks consistían en agregar líneas de código en el paquete para que los nodos los ejecutara, pero ellos no incorporan elementos de control mediante el software en los elementos de la conmutación, como lo es SDN.

Otra iniciativa tuvo lugar en la década de 1990 fue “Devolver Control of ATM Network” (DCAN). Este proyecto tenía como objeto diseñar y desarrollar la infraestructura necesaria para el control y gestión escalable de redes de cajeros automáticos. se basaba en que las funciones de control y gestión de muchos dispositivos se deben otorgar a una entidad externa dedicada solo a ello, básicamente este es el concepto de SDN. Además, DCAN propone un protocolo minimalista entre el gestor y la red, lo que equivalente a OpenFlow.

En el 2006 se propuso NETCONF como protocolo de gestión para modificar la configuración de los dispositivos de red. El protocolo permitía a los dispositivos de res exponer un API a través de la cual el dato de configuración extensibles podía ser enviado y recuperados. esta propuesta cumplida con el objetivo de la reconfiguración del dispositivo y construcción para la gestión, aunque esto no separa el plano de control y el plano de datos además esta no se considera una red totalmente programable ya que ayuda a la configuración automatizada pero no el control directo del estado ni permite un rápido despliegue de servicios y aplicaciones innovadores.

Para el mismo surge la propuesta que antecede inmediatamente a OpenFlow llama proyecto SANE/Ethane, quien define una arquitectura para las redes empresariales. Ethane da

una propuesta muy similar a SDN la cual emplea dos componentes: (1) un controlador para decidir si un paquete debe ser enviado (2) Switch Ethane que consiste en una tabla de flujo y un canal seguro al controlador. Siendo así Ethane el que sentó las bases para lo que se convertiría en Redes definidas por software.

En el 2007 se dio comienzo a la creación de SDN realizado en la universidad de Stanford por los profesores Nick McKeown, Scott Shenker y el estudiante de doctorado Martín Casado desarrollaron OpenFlow y fundaron Nicira, una compañía de virtualización de redes. Es a partir de este momento donde se sitúa el nacimiento de SDN.

En resumen, la historia se divide en tres etapas con diferentes contribuciones.

Redes Activas. Desde mediados de la década de 1990 hasta los principios de la década de 2000, se introdujeron funciones programables en las redes, llevando una mayor innovación

Separación de datos de control y datos. Alrededor del 2001 al 2007), Se desarrollan interfaces abiertas entre los planos de control y de datos.

APIOpenFlow y sistemas operativos de red. Del 2007 al 2010, Adopción de la interfaz abierta de formas desarrolladas para hacer la separación del plano de control y datos escalable y práctico, también se da la virtualización de las redes lo cual juega un papel importante a lo largo de la evolución histórica del SDN.

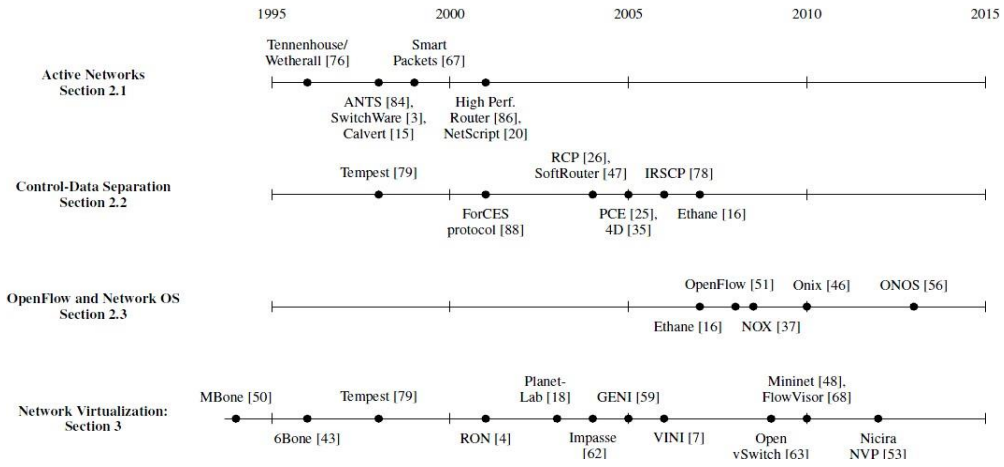

Figura 1. Línea de tiempo de las SDN desde 1995 hasta el 2015.

Fuentes: Feamster, N., Rexford, J., & Zegura, E.W. (2014). The road to SDN: an intellectual history of programmable networks. Computer Communication Review, 44, 87-98

A Partir del 2008 investigadores de la universidad de Stanford Nick McKeown , idearon Openflow, una nueva forma en la que experimentar en las redes actuales, pero no se define relación entre SDN y OpenFlow, la concepción de un software de control separado de la red fue visto como una implementación real de SDN, estandarizando la forma en la que el controlador se comunica con los dispositivos, permitiendo la programación de las tablas de flujos por parte de las aplicaciones software y especificando cómo migrar el control de la red al controlador.

Se establecen versiones de OpenFlow a través del tiempo mediante un grupo colaborativo de universitarios y administradores de red liderado por miembros de Stanford y Berkeley esto dio como resultado versiones experimentales desde el 2007 con 0.1.0 y en el 2008 con 0.2.0, 0.8.0, 0.8.1,0.8.2, 0.8.9 y la versión 0.9 .0 a mediados del 2009, pero en diciembre del mismo año se lanzó la versión 1.0.0, la más ampliamente adoptada.

Las tecnologías de las redes definidas por software se han estudiado en diferentes partes del mundo ocasionando un gran auge por el conocimiento de esta tecnología, en España se lleva a cabo un estudio de las SDN por medio del protocolo OpenFlow ejercido por David Andrés serrano carrera en el 2015, estudiando su arquitectura y los desencadenes para su implementación. En Ecuador por su parte la universidad de las fuerzas armadas ESPE, con los estudiantes Alba Ruiz, y Dario Xavier, realizaron trabajos de diseño e implementación de un prototipo de SDN, comparando los diferentes controladores y lenguajes de programación que ellos utilizan y buscando la mejor disponibilidad y optimización de la comunidad de red.

A nivel nacional en Bogotá la universidad Santo Tomás y los estudiantes Julián Camilo Sombredero Alonso y Erick Ferney Silva Herrera, realizaron análisis comparativo de prestaciones entre SDN y redes IP convencionales, haciendo comparaciones de Jitter y Delay en escenarios físicos, simulado y emulado, con respecto a protocolos y direccionamiento. Además, en la universidad Pontificia se realiza una implementación de un controlador Openflow para el manejo de switches OpenFlow a cargo del estudiante Carlos Alberto Contreras Pardo, donde se prueba diferentes controladores en tiempos y manejo de flujos.

A nivel regional se encuentra que la universidad Francisco de Paula Santander Ocaña con los estudiantes Oscar Ignacio Lobo y Darwin Acosta, proponen un diseño de una arquitectura basada en tecnología “sdn” (redes definidas por software) para el laboratorio de redes y telecomunicaciones de la Universidad Francisco de Paula Santander Ocaña, donde destacan los beneficios y usos de las redes definidas por software.

2.2 Marco contextual.

Laboratorio de redes y telecomunicaciones (LRYT)" Laboratorio de redes y telecomunicaciones" ubicado en la Universidad Francisco de Paula Santander Ocaña.

El laboratorio cuenta con dispositivos físicos como switch y routers de la marca CISCO y un servidor de la marca Hewlett Packard las cuales son empresas reconocidas en el mercado y de alta calidad para las redes de información, dicho servidor tiene instalados los sistemas operativos Windows server 2000 y Ubuntu 16.4 LT Donde este último será utilizado para los casos de usos utilizando las redes definidas por software.

Cabe destacar que el Laboratorio de redes y telecomunicaciones está disponible para todos los estudiantes pertenecientes a las carreras de ingeniería de sistemas y técnicos en telecomunicaciones y dicho proyecto beneficiará a tal comunidad.

2.3 Marco conceptual

Redes definidas por software (SDN). Según la Open Network Foundation (ONF), "Software-Defined Networking (SDN) es una arquitectura emergente que es dinámica, manejable, rentable y adaptable, haciéndola ideal para la naturaleza dinámica de banda ancha de las aplicaciones actuales. Esta arquitectura desvincula las funciones de control y reenvío de la red permitiendo al control de la red hacerse programable directamente quedando abstraída la infraestructura subyacente para las aplicaciones y los servicios de red. El protocolo OpenFlow ™

es un elemento fundamental para la construcción de soluciones SDN". (Open Networking Foundation, 2018).

OpenFlow. Según la ONF, “OpenFlow® es la primera interfaz de comunicaciones estándar definida entre las capas de control y de reenvío de una arquitectura SDN. OpenFlow® permite el acceso directo y la manipulación del plano de reenvío de dispositivos de red, tales como switches y enrutadores, tanto físicos como virtuales (basados en hipervisor)”. (Open Networking Foundation, 2018)

POX. plataforma de desarrollo de código abierto para aplicaciones de control de redes definidas por software (SDN) basadas en Python, como los controladores SDN de OpenFlow. POX, que permite el desarrollo rápido y la creación de prototipos, es cada vez más comúnmente utilizado que NOX, un proyecto hermano. (Rouse, SearchSDN, 2013)

Beacon. “es un controlador OpenFlow, rápido, multiplataforma basado en java, que soporta operaciones basadas en eventos y operaciones con atreves de hilos.” (Erickso, Home Beacon Confluence, 2013)

Controlador SDN. “Un controlador SDN es una aplicación de red definida por software (SDN) que gestiona el control de flujo para habilitar la creación de redes inteligentes. Los controladores SDN se basan en protocolos, como OpenFlow, que permiten a los servidores indicar a los switches dónde enviar paquetes.”. (Rouse, What is SDN controller (software-defined networking controller), 2012)

Virtualización de red. “refiere a la capacidad de proporcionar redes de extremo a extremo que se abstrae de los detalles de la red física subyacente de una manera similar a cómo la virtualización de servidores proporciona recursos informáticos que son abstraídos de los detalles de los servidores subyacentes basados en x86”. (Citrix Systems, Inc, 2014)

Switch. “dispositivo que canaliza los datos entrantes de cualquiera de los múltiples puertos de entrada al puerto de salida específico que llevará los datos hacia su destino previsto**”** (Rouse, Search Networking, 2017)**.**

Router. Dispositivo o, en algunos casos, un software de una computadora, que determina la mejor manera de enviar un paquete a su destino. (Rouse, SearchNetworking, 2016)

Servidor. “Un servidor es un programa informático que proporciona servicios a otros programas informáticos (y sus usuarios) en el mismo o en otros ordenadores. El equipo en el que se ejecuta un programa de servidor también se conoce con frecuencia como un servidor. Esa máquina puede ser un servidor dedicado o usado para otros propósitos también”. (Rouse, techtarget, 2018)

Virtual Switch. “Un conmutador virtual es un programa de software que permite que una máquina virtual (VM) se comunique con otra. Al igual que su contraparte, el conmutador físico Ethernet, un conmutador virtual hace más que simplemente enviar paquetes de datos. Puede dirigir inteligentemente la comunicación en la red inspeccionando los paquetes antes de pasarlos. Algunos vendedores incorporan interruptores virtuales directamente en su software de

virtualización, pero también se puede incluir un conmutador virtual en el hardware de un servidor como parte de su firmware". (Rouse, techtarget, 2010)

Mininet. Software estandarizado por la OFN para las emulaciones de un entorno SDN, el cual crea una red virtual realista, ejecutando kernel real, switch y código de aplicación, en una sola máquina.

2.4 Marco teórico

Basado en los estudios y descubrimientos hechos por la universidad de Stanford en el 2007 se fundó Nicira una empresa que desarrolló versiones sobre OpenFlow y open Vswitch aunque la empresa se cerró los estudios, se han expuesto teorías y han surgido actualizaciones en los protocolos.

Las redes definidas por software son una conexión de nodos (switch) interconectados formando una gráfica, dada su función G= V (V, E), donde E describe la conectividad entre los switch.

La estructura planteada para las aplicaciones SDN se define de la siguiente manera:

- La capa de infraestructura
- Controlador Openflow
- Capa de control
- API
- Capa de aplicación.

Como se muestra en la siguiente figura.

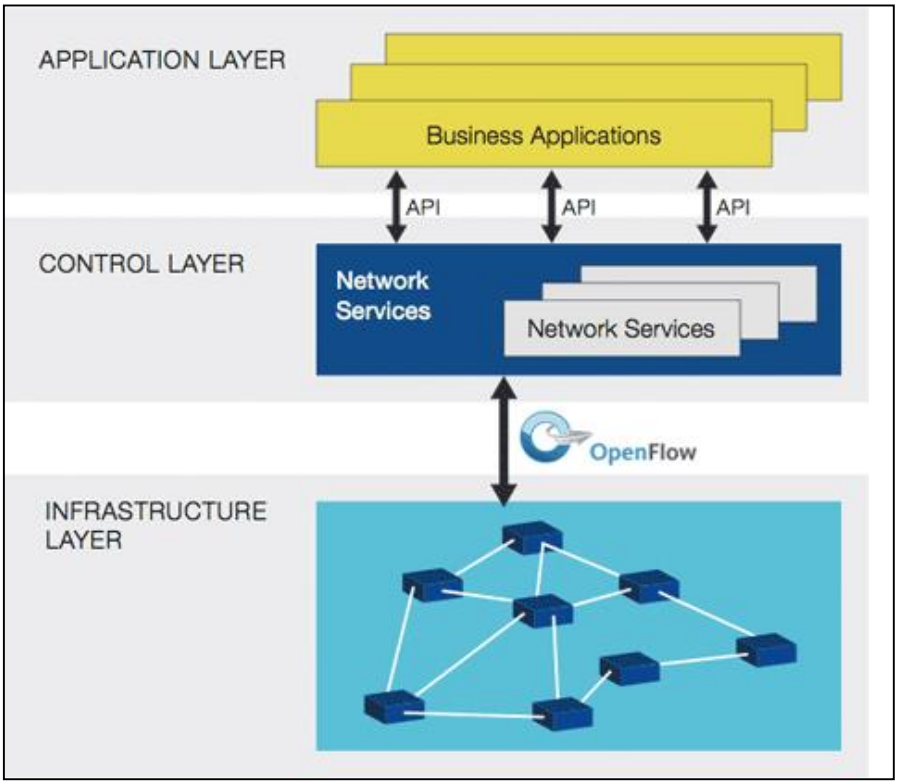

Figura 2. Arquitectura de una SDN

Fuente:OpenNetworkFundation:theNewNormForNetworks,https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf

Basados en la infraestructura de las redes definidas por software se encuentra Openflow, para el cualse han creado protocolos para su aplicación. Muchos de estos protocolos son de carácter único y se plantean diferentes teorías de ellos, a su vez nacen diferentes controladores a través de la historia, como lo son NOX, POX, Beacon entre otros (Open Network Fundation, 2012) . Como se muestra en la imagen es la arquitectura básica de las SDN, sus componentes arquitectónicos e interacciones. A su vez se muestra dicha arquitectura está compuesta por tres capas principales: infraestructura, control, aplicaciones. Aunque la inteligencia de la red está basada en los controladores hechos en software que mantienen la visión global de la misma.

En teoría los Switch deben delegar su inteligencia al controlador y pasar a ser unidades de conmutación de tráfico, es allí donde se implantan las funcionalidades a través de la interfaz de programación de aplicaciones (Application Programing Interface API).

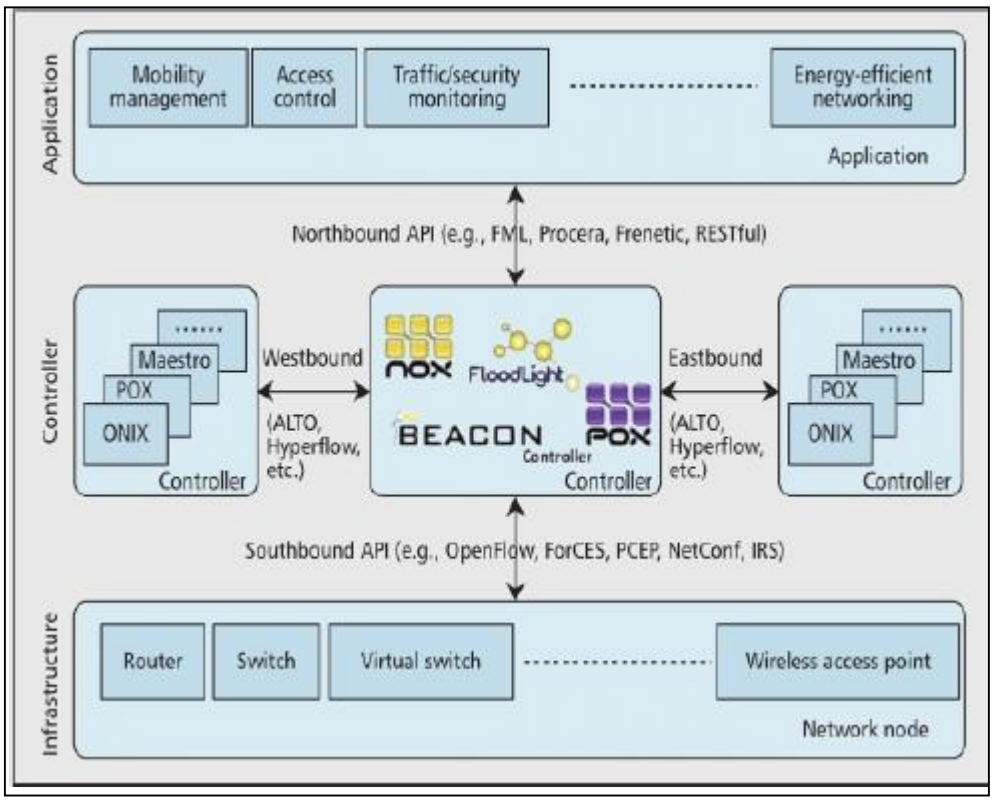

Figura 3. Estructura funcional del SDN

Fuente Ominike, Akpovi & Seun, Ebiesuwa & A. O., Adebayo & Osisanwo, F. (2016). Introduction to Software Defined Networks (SDN). International Journal of Applied Information Systems. 11. 10-14. 10.5120/ijais2016451623.

En el Nacimiento de Openflow se establecen ciertas relaciones a considerar. Un conmutador OpenFlow aprovecha que los Switch y los router ethernet contienen tablas de flujo, para las funciones como cortafuegos, Qos o estadísticas (Ominike, 2016). Dependiendo el fabricante del dispositivo la tabla es diferente, pero estas mantienen conjuntos de funciones comunes. En el dispositivo OpenFlow se disponen de al menos tres partes:

Tablas de flujos: cada tabla contiene campos de búsqueda de coincidencias de los paquetes entrantes, las cuales definen que hacer con dicho paquete.

Canal seguro: Conectar el dispositivo al controlador, permite el envío tanto de comando y de paquetes mediante el protocolo OpenFlow.

Protocolo OpenFlow: Es un estándar de comunicación entre el conmutador y el controlador, además permite que este ultime realiza inserciones, modificaciones, eliminación y búsquedas de las entradas de flujos de la tabla a través del canal seguro.

El funcionamiento del controlador es de la siguiente manera:

Los campos de las cabeceras son extraídos para buscar coincidencias.

Se comparan con las reglas definidas para entrar en la tabla de flujos de OpenFlow, donde se tiene en cuenta el orden descendente de prioridad. un campo en la tabla de flujos puede tener valor “ANY” el cual hace coincidencia con todos los paquetes.

Si hay una coincidencia, las acciones son especificadas de esa entrada se realizan sobre el paquete si no los primeros 200 bytes del paquete son enviados al controlador para ser procesado.

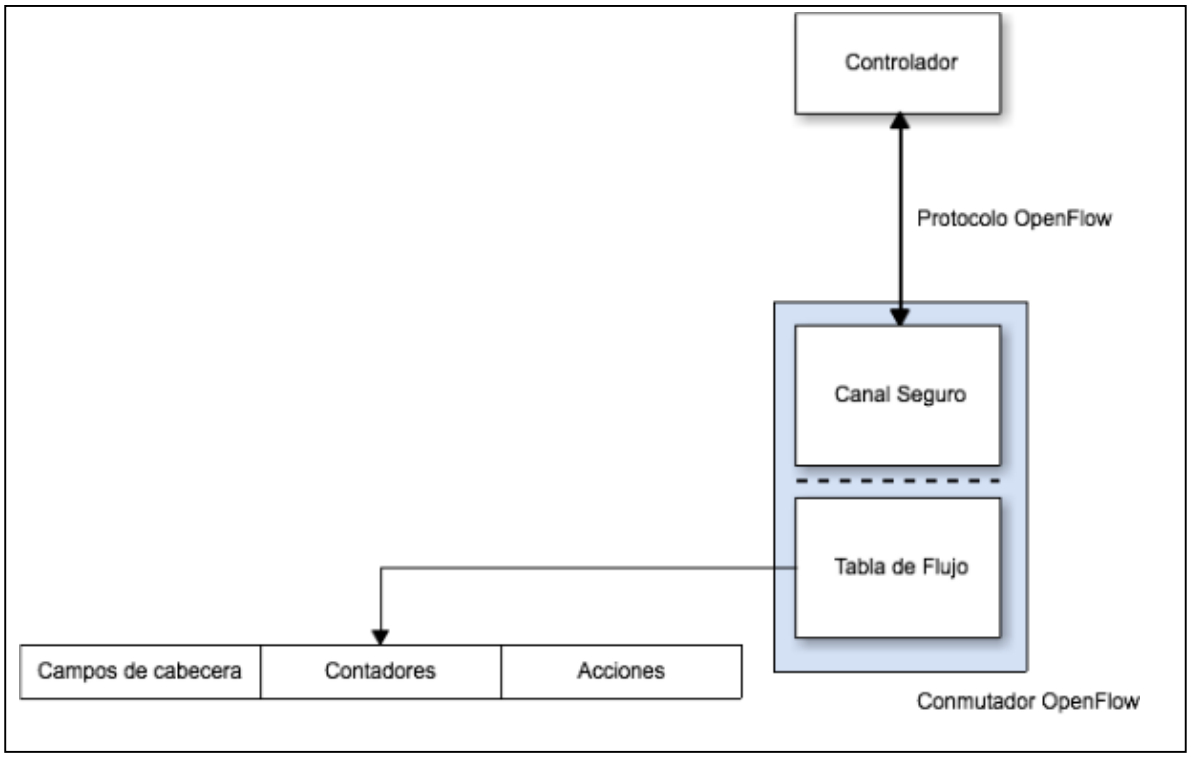

Figura 4.Funcionamiento de SDN

Entre las versiones de OpenFlow switch se encuentran:

Versión 1.2.0: incluye soporte a las direcciones IPv6, así como la posibilidad de conectar un conmutador a múltiples controladores, que puede comunicarse entre ellas.

Versión 1.3.0: se puede controlar la tasa de paquetes mediante medidores de flujo. Se añaden conexiones auxiliares entre el conmutador y el controlador y se incorporan cookies y campos de duración en las estadísticas.

Versión 1.4.0 y 1.5.0: se añaden características como monitorización de flujos, “pipeline” sensibles a paquetes y estadísticas extensibles en las entradas de las tablas de flujo

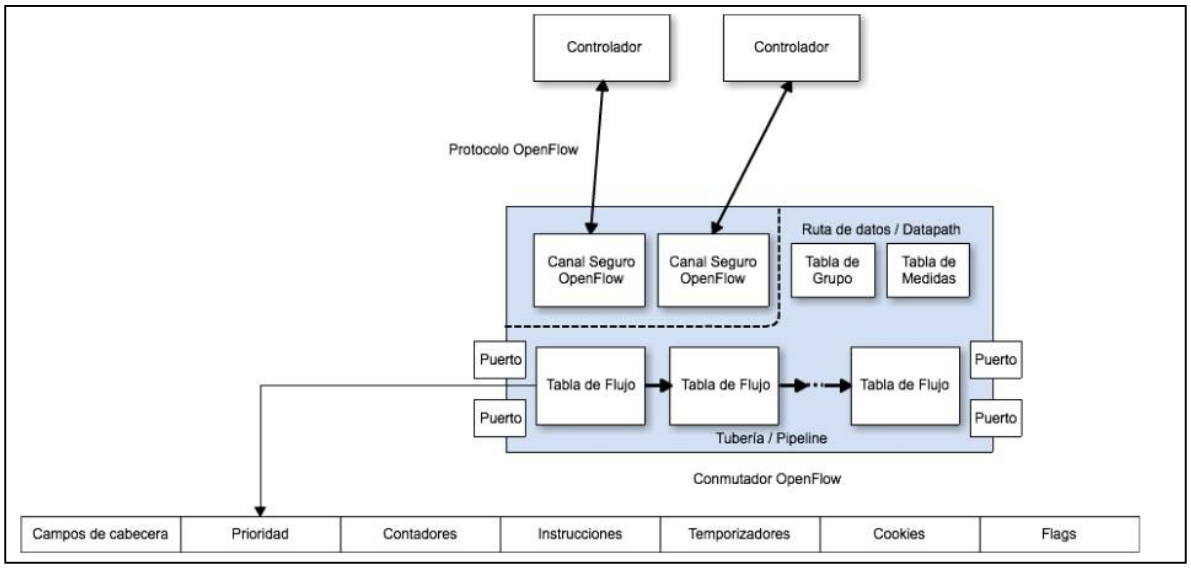

Figura 5.Funcionamiento SDN última versión

Las tablas de flujo para los Switch OpenFlow han sufrido diferentes cambios, desde su primera versión 1.0 hasta la versión 1.5, donde puede observar que se han establecido diferentes mejoras, ya sea de usabilidad como de seguridad.

Tabla 1

Tabla de flujo OpenFlow 1.0

Header Filds	Counters	Actions

Nota: Recuperado de Open Network Fundations, Open Flow Specifications, Diciembre 31 2009

Header Filds para unir contra paquetes, (Open Networking Foundation, 2009)

Counters contadores para actualizar para el paquete coincidente (Open Networking Foundation, 2009)

Actions para aplicar a paquetes coincidentes (Open Networking Foundation, 2009)

Donde se establecen unas cabeceras de la siguiente forma para las Actions:

Tabla 2

Campos de paquetes utilizados para coincidir con las entradas de flujo.

Ingress Port	Ether Source	Ether dst	Ether type	VLAN id	Vlan priority	IP src	IP dst	IP proto	IP ToS bits	TCP/UDP src port	TCP/UDP dst port

Nota. Recuperado de Open Network Fundations, Open Flow Specifications, Diciembre 31 2009

En su última versión se actualiza su tabla de flujo quedando de la siguiente manera:

Tabla 3

Componentes principales

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie	Flags

Nota. Recuperado de Open Network Fundations, Open Flow Specifications, Marzo 26 2015.

Campos coincidentes: para unir contra paquetes. Estos consisten en el puerto de entrada y los encabezados de paquetes y opcionalmente otros campos de canalización como los metadatos especificados por una tabla anterior. (The Open Networking Foundation, 2015).

Prioridad: coincidencia de precedencia de la entrada de flujo. (The Open Networking Foundation, 2015)

Contadores: actualizado cuando se combinan los paquetes. (The Open Networking Foundation, 2015).

Instrucciones: para modificar el conjunto de acciones o el procesamiento de la tubería. (The Open Networking Foundation, 2015)

Tiempos de espera: cantidad máxima de tiempo o tiempo de inactividad antes de que el flujo expire con el interruptor. (The Open Networking Foundation, 2015)

Cookie: valor de datos opaco elegido por el controlador. Puede ser utilizado por el controlador para filtrar el flujo entradas afectadas por estadísticas de flujo, modificación de flujo y solicitudes de eliminación de flujo. No usado cuando procesamiento de paquetes. (The Open Networking Foundation, 2015)

Banderas: las banderas alteran la forma en que se administran las entradas de flujo (The Open Networking Foundation, 2015)

Se encuentran diversas herramientas de simulación y emulación para las SDN como lo son STINET, NS-3, MININET.

El emulador Mininet es una plataforma de pruebas de red de código abierto y rápidamente configurable. Hasta ahora, es la herramienta de apoyo a la investigación de las SDN OpenFlow más conocida, como se denota en la conferencia ONS 2013. Mininet utiliza hosts virtuales, switches y enlaces para crear una red en un solo núcleo del sistema operativo, y utiliza la pila de red real para procesar paquetes y conectarse a las redes reales. Además, las aplicaciones de red basadas en Unix/Linux, también se pueden ejecutar en los hosts virtuales. En una red OpenFlow emulada por Mininet, una aplicación de controlador real OpenFlow se puede ejecutar en una máquina externa o en el mismo equipo en el que se emulan los hosts virtuales.

2.5 Marco legal

Ley 873 de 2004. Por medio de la cual se aprueban el Instrumento de Enmienda a la Constitución de la Unión Internacional de Telecomunicaciones, con las enmiendas adoptadas por la Conferencia de plenipotenciarios, Enmiendas adoptadas por la Conferencia de Plenipotenciarios, firmado en Minneápolis, el seis (6) de noviembre de mil novecientos noventa y ocho (1998), y el Instrumento de Enmienda al Convenio de la Unión Internacional de Telecomunicaciones, con las enmiendas adoptadas por la Conferencia de Plenipotenciarios, Enmiendas adoptadas por la Conferencia de Plenipotenciarios, firmado en Minneápolis, el seis (6) de noviembre de mil novecientos noventa y ocho (1998).

Artículos 12 y 68 Reglamentados por el Decreto 2044 del 19 de septiembre de 2013. Ley reglamentada por el Decreto 2693 del 21 de diciembre de 2012. Parágrafo 2° del artículo 57 modificado por el artículo 59 de la Ley1450 de 2011, Inciso 1°y 3° y el parágrafo 1°y 2° del artículo 69 derogado por el artículo 276 de la Ley 1450 de 2011.numerales 6 y 7 del artículo 18, el numeral 11, del artículo 28 y el artículo 29 de la Ley 1341 de 2009 derogados por el Decreto 4169 de 2011. (Secretaría Jurídica Distrital de la Alcaldía Mayor de Bogotá D.C., 2009)

Ley 1341 del 30 de julio del 2009. Define los principios y conceptos sobre la sociedad de la información y la organización de las tecnologías de la información y las comunidades, crea la agencia nacional de espectro se dictan otras disposiciones tales como Disposiciones generales, Provisión de las redes y servicios y acceso a recursos escasos, Organización institucional , Promoción al acceso y uso de las tecnologías de la información y las

comunicaciones, reglas de solución de controversias en materia de interconexión, régimen de protección al usuario, régimen de los proveedores de redes y servicios de las tecnologías de información y las comunicación, Régimen de infracciones y sanciones, régimen de transición y Disposiciones finales (Ministerio de tecnologías de la información y las comunicaciones, 2009) .

La ONF ha propuesto ítems de especificaciones para las SDN, Incluye todas las normas que definen un protocolo, modelo de información, funcionalidad de los componentes y documentos relacionados con el marco. Es esta categoría de Especificación Técnica la que se identifica como tal porque es una publicación normativa que tiene la política y licencias de la ONF sobre DPI guiando su uso posterior. (Open Networking, 2018) dichas especificaciones se encuentran disponibles directamente en su web junto con recomendaciones y publicaciones, a su vez se interpreta como una forma de trabajar propuesta por dicha fundación.

Capítulo 3. Diseño metodológico

3.1 Tipo de investigación

El siguiente proyecto se basa en una investigación experimental teniendo en cuenta que la manipulación de las variables y casos planteados para el estudio van hacer en entornos controlados y se efectuarán procedimientos concretos y se observará el cambio que puedan surgir en las variables como velocidad, seguridad, calidad de servicio e integridad de datos.

Por otra parte, tendrá un enfoque de tipo cuantitativo teniendo en cuenta que se pretende analizar parámetros como la velocidad y asertividad en la entrega de paquetes, que pueden representarse en valores numéricos y sobre los cuales se puede llevar una evaluación cuantitativa.

3.2 Población y muestra

3.2.1 Población. Teniendo en cuenta la naturaleza de la investigación se tiene una población de doscientos ochenta y cuatro (284) Estudiantes pertenecientes a la carrera de Ingeniería de sistemas en el noveno y décimo semestre de la Universidad Francisco de Paula Santander Ocaña.

3.2.2 Muestra. Dada que en el proyecto se muestra una población pequeña para dichos estudios, se toma como muestra el cien por ciento (100%) de dicha población que equivale a los 284 estudiantes tomados de la población.

3.3 Técnicas e instrumentos de recolección de información

Se utilizarán los siguientes instrumentos de recolección de información:

Observación y Análisis: Las cuales se realizarán a través del comportamiento de las SDN en sus variables y distintos entornos.

Revisión: Se realiza una revisión literaria de las diferentes aplicaciones de las SDN y los resultados obtenidos.

Capítulo 4. Presentación de resultados

Se encuentran muchos controladores para las SDN, pero no todas son de carácter libre a continuación se describirán algunas características de controladores libres en el mercado.

4.1 NOX

Una red NOX se basa en un conjunto de conmutadores y servidores conectados en red. NOX se ejecuta en dichos servidores, las aplicaciones que este utiliza son para la toma de decisiones de gestión, para controlar el tráfico debe manipular interruptores. NOX intercambia la escalabilidad contra la flexibilidad, y ambos son cruciales para grandes redes empresariales con diversos requisitos. Las vistas de redes NOX incluye el nivel de los switch en la topología, las ubicaciones de usuarios, hosts, entre otros elementos de la red además de los servicios (por ejemplo, HTTP oNFS) que se ofrecen. La vista incluye todos los enlaces entre nombres y direcciones, pero no incluye el estado actual del tráfico de red. (Yanhe Liu, 2013)

4.2 POX

Controlador de SDN desarrollado en código abierto para aplicaciones SDN basado en Python, permite el desarrollo rápido de creación de prototipos, debido a su gran aplicación más que su proyecto hermano NOX es considera una mejor de controlador anteriormente mencionado.

Oficialmente para funcionar POX requiere de la versión python 2.7 en adelante es ejecutable en Linux, Mac Os y windows, aunque se ha utilizado en otros sistemas estos son los oficiales, como la mayoría de los desarrollos se realizan en Mac Os en algunos casos estos no funcionan en los otros sistemas operativos (SO), dicho esto el tiempo que se tarda una aplicación en solucionar problemas planteados depende de los reportes de dichos problemas, en general los problemas presentados por este controlador en los diferentes SO como Linux se notan con gran rapidez esencialmente cuando son problemas de gran escala, y con gran lentitud en Windows. Además, se comunica fácilmente con switches Openflow 1.0 e incluye soporte especial para extensiones como vSwitch / Nicira. pox.py arranca POX. Toma una lista de nombres de módulos en la línea de comando, localiza los módulos, llama a su función de inicio () (si existe) y luego transita al estado "arriba". POX se puede usar con el intérprete de Python "estándar" (CPython), pero también es compatible con PyPy. (Ali Al-Shabibi, 2013)

4.3 Beacon

Es un controlador OpenFlow basado en Java. debido a esto es multiplataforma y modular que permite operaciones basadas en eventos y subprocesos.Las características principales se Beacon se basa en:

Estabilidad: Se desarrolla desde principios del 2010, y se ha utilizado en varios proyectos de investigación, clases de redes y despliegues de prueba. Actualmente este desarrollo suministra un centro de datos experimental de conmutación de 100 Vswitch y 20 Switch físicos los cuales se han ejecutado durante meses sin tiempo de inactividad.

Multiplataforma: Gracias a su desarrollo en Java permite ejecutarse en diferentes dispositivos desde servidores multinúcleo de alta gama de Linux hasta teléfonos Android.

Código abierto: Se basa en una licencia GPL v2 y la excepción de licencia FOSS de la universidad de Stanford v1.0.

Dinámico: Los paquetes de código en beacon puede, iniciarse, detenerse, actualizarse, e instalarse en tiempo de ejecución, sin interrumpir otros paquetes no dependientes.

Desarrollo rápido: Es multihilo.

Web UI: Beacon integra opcionalmente el servidor web empresarial de Jetty y un marco de interfaz de usuario extensible.

Frameworks: Beacon se basa en marcos Java maduros como Spring y equinox (OSGi)

(Erickso, Home Beacon Confluence, 2012)

4.4. Trema

Trema es un framework de código abierto para desarrollar controladores OpenFlow para SDN programados en los lenguajes Ruby y C. NEC creó la tecnología y la utiliza como base de su controlador programable de flujo comercial.

Trema es un marco de programación completa que permite a los usuarios desarrollar y probar controladores OpenFlow en una computadora portátil. Los desarrolladores pueden crear

sus propios controladores basados en Trema al agregar Script de mensajería además maneja las versiones de OpenFlow 1.3.

Características de Trema:

Desarrollador: Trema es un framework Full-stack OpenFlow para Ruby/c, orientado a desarrolladores OpenFlow

Licencia: Es un software libre desarrollado bajo la licencia GPLv2 con una comunidad abierta.

Uso: Trema es de fácil uso, además incluye un ambiente de desarrollo con una cadena herramientas integradas (Trema, 2011)

4.5 RYU

RYU es un marco de red de software definido por componentes. Ryu proporciona componentes de software con una API bien definida que facilita a los desarrolladores la creación de nuevas aplicaciones de administración y control de redes. Ryu admite varios protocolos para administrar dispositivos de red, como OpenFlow, Netconf, OF-config, etc. Acerca de OpenFlow, Ryu es compatible con las versiones 1.0, 1.2, 1.3, 1.4, 1.5 y Nicira. Todo el código está disponible de forma gratuita bajo la licencia Apache 2.0.

El código fuente del controlador Ryu está alojado en GitHub y administrado y mantenido por la comunidad abierta de Ryu. OpenStack, que ejecuta una colaboración abierta centrada en el desarrollo de un sistema operativo en la nube que puede controlar los recursos de cómputo,

almacenamiento y redes de una organización, admite implementaciones de Ryu como controlador de red. (Community, 2017),

Debido a estas definiciones se pueden concluir varios aspectos *(ver tabla4)*, con los cuales podemos establecer las diferencias que se encuentran en cada uno de los controladores y cuál de ellos nos posibilita una mejor adaptación a nuestro proyecto.

Tabla 4

Diferenciación de controladores.

	NOX	**POX**	**BEACON**	**TREMA**	**RYU**
OpenFlow	OF v1.0, y OF v.1.3 versión actualizada	OF v1.0	OF v1.0	OF v1.3	OF v1.0,v1.2,v1.3,v 1.4,v1.5, Nicira
Virtualización	Mininet y OpenvSwitc h	Mininet y OpenvSwitch	Mininet y OpenvSwitch	Construcción de una herramienta de virtual	Mininet y OpenvSwitch
Lenguaje de desarrollo	C++	Python	Java	Ruby/C	Python
Interfaz Gráfica	Python+, Qt4	Python+, Qt4, Web	Web	No	Web
Soporte de plataformas	Linux	Linux, Mac OS, Windows	Linux, Mac OS, Windows y Android para móviles	Linux	Linux
Soporte de OpenStack	No	No	No	Si	Si
Código Abierto	Si	Si	Si	Si	Si
Documentación	Buena	Media	Pobre	Media - alta	Media - alta

Nota. Fuente Autor del proyecto

4.6 Mininet

Mininet crea una red virtual realista, que ejecuta kernel real, switch y código de aplicación, en una sola máquina (VM, nube o nativo), en segundos.

Debido a que puede interactuar fácilmente con su red mediante la CLI (y la API) de Mininet, personalizarla, compartirla con otros o implementar en hardware real, Mininet es útil para el desarrollo, la enseñanza y la investigación.

Mininet también es una excelente forma de desarrollar, compartir y experimentar con OpenFlow y sistemas de redes definidas por software.
Mininet se desarrolla y respalda activamente, y se emite bajo una permisiva licencia de Open Source de BSD.

4.7 Open VSwitch

Es un conmutador virtual multicapa de calidad de producción bajo la licencia de Apache 2.0. Esta herramienta fue diseñada para permitir la automatización masiva de la red a través de la extensión programática, al tiempo que respalda las interfaces y protocolos de gestión estándar.

Es el conmutador predeterminado en XenServer 6.0, Xen Cloud Platform y también es compatible con Xen, KVM, Proxmox VE y VirtualBox. También se ha integrado en muchos

sistemas de gestión virtual, incluidos OpenStack, openQRM, OpenNebula y oVirt. El kernel datapath se distribuye con Linux, y los paquetes están disponibles para Ubuntu, Debian, Fedora y openSUSE. Open vSwitch también es compatible con FreeBSD y NetBSD. La versión Open vSwitch en desarrollo se ha trasladado a DPDK.

4.8 OpenDaylight

Es un proyecto SDN de código abierto destinado a mejorar las SDN al ofrecer un marco comunitario y compatible con la industria para el Controlador OpenDaylight, que ha cambiado su nombre por la Plataforma OpenDaylight. Está abierto para cualquier persona, incluidos los usuarios finales y los clientes, y proporciona una plataforma compartida para aquellos con objetivos de SDN para trabajar juntos para encontrar nuevas soluciones.

Bajo la Fundación Linux, OpenDaylight incluye soporte para el protocolo OpenFlow, pero también puede admitir otros estándares SDN abiertos.

El protocolo OpenFlow, considerado el primer estándar SDN, define el protocolo de comunicaciones abierto que permite que el controlador SDN trabaje con el avión de reenvío y realice cambios en la red. Esto le da a las empresas la capacidad de adaptarse mejor a sus necesidades cambiantes y tener un mayor control sobre sus redes.

El controlador OpenDaylight puede implementarse en una variedad de entornos de red de producción. Puede admitir un marco de controlador modular, pero puede proporcionar soporte para otros estándares SDN y protocolos futuros.

Así mismo, OpenDaylight expone las API abiertas hacia el norte, que son utilizadas por las aplicaciones. Estas aplicaciones usan el Controlador para recopilar información sobre la red, ejecutar algoritmos para realizar análisis y luego usar el Controlador OpenDaylight para crear nuevas reglas en toda la red.

El controlador OpenDaylight se implementa únicamente en software y se mantiene dentro de su propia máquina virtual Java (JVM). Esto significa que puede implementarse en plataformas de hardware y sistema operativo que admiten Java. Para obtener los mejores resultados, se sugiere que OpenDaylight Controller use una distribución de Linux reciente y, al menos, Java Virtual Machine 1.7. (OpenDaylight Project a Series of LF Projects, LLC, 2018)

4.9 HP Virtual Application Networks SDN Controller

HP Virtual Application Networks (VAN) El software del controlador SDN proporciona un punto de control unificado en una red habilitada para OpenFlow, simplificando la administración, el aprovisionamiento y la orquestación. Esto permite la entrega de una nueva generación de servicios de red basados en aplicaciones. También proporciona interfaces de programa de aplicaciones (API) abiertas para permitir a los desarrolladores de terceros ofrecer soluciones innovadoras para vincular dinámicamente los requisitos comerciales a la infraestructura de red a

través de programas Java personalizados o interfaces de control RESTFULL de propósito general. El controlador VAN SDN está diseñado para funcionar en campus, centros de datos o entornos de proveedores de servicios.

Características:

Plataforma de clase empresarial para la entrega de una amplia gama de innovaciones de red.

Soporta protocolos OpenFlow 1.0 y 1.3

Soporte para más de 50 modelos de conmutadores HP habilitados para OpenFlow

Habilita el desarrollo de aplicaciones SDN de terceros.

Arquitectura de controlador extensible, escalable y flexible

4.10 GNS3

GNS3 es utilizado por cientos de miles de ingenieros de redes en todo el mundo para emular, configurar, probar y solucionar problemas de redes virtuales y reales. GNS3 le permite ejecutar una pequeña topología que consta de solo unos pocos dispositivos en su computadora portátil, a aquellos que tienen muchos dispositivos alojados en múltiples servidores o incluso alojados en la nube.

GNS3 consta de dos componentes de software:

El software GNS3 todo en uno (GUI)

La máquina virtual (VM) GNS3

(David Bombal, 2018)

Acorde con la investigación y con los requerimientos del proyecto, algunos de estos controladores nos son de utilidad y otros no, debido a los protocolos soportados, las herramientas con las que trabajan, su documentación, entre otras cosas. RYU es el controlador que se decide utilizar*(ver tabla 4),* ya que consta con herramientas de virtualización, soporta el protocolo OpenFlow 1.3 en adelante, tiene una documentación amplia, entre otras cosas, además, este controlador actúa como framework, haciendo así que sea posible la utilización del mismo para la generación de controladores propios adecuándolos a los requerimientos que pueda tener una empresa.

4.11 Instalación de herramientas y controladores

Se mostrará el proceso de instalación que se llevó acabo de las herramientas necesarias para la ejecución del proyecto, además puede dirigirse al *Apéndice 1*, el cual mostrara la instalación de otros controladores que nos ayudaron con el análisis de ellos mismos.

4.11.1 Instalación Mininet. Se inicia con la instalación de Mininet el cual tiene diferentes modos, como lo son por máquina virtual, instalación nativa e instalación por paquetes. En este proyecto se realizará la instalación nativa la cual se hará de la siguiente manera:

1. Clonación del repositorio en Git.

git clone git://github.com/mininet/mininet

NOTA: Recordar tener previamente instalado Git en su sistema operativa, en caso contrario instalarlo con apt install git. Si se desea descargar una versión en especial se debe indicar explícitamente.

2. Ejecutar el instalador de Mininet.

cd mininet

git tag # list available versions

git checkout -b x.x.x x.x.x

cd..

NOTA: las X anteriores significan las versiones que se pueden instalar. por ejemplo 2.2.1 2.2.1

3. Instalación de paquetes de Mininet.

Para la instalación de los paquetes hay diferentes opciones como lo es -a, -nfv y -s mydir pero en este caso se instalarán todas de la siguiente manera.

mininet/util/install.sh

Además, si es necesario instalar software adicional de Openflow se pueden buscar con el comando install.sh -h esto podría incluir wireshark si no está incluido en la versión de Mininet

Para probar la instalación de Mininet puede ejecutar el comando **sudo mn --testpingall** el cual creará una red y un controlador virtual

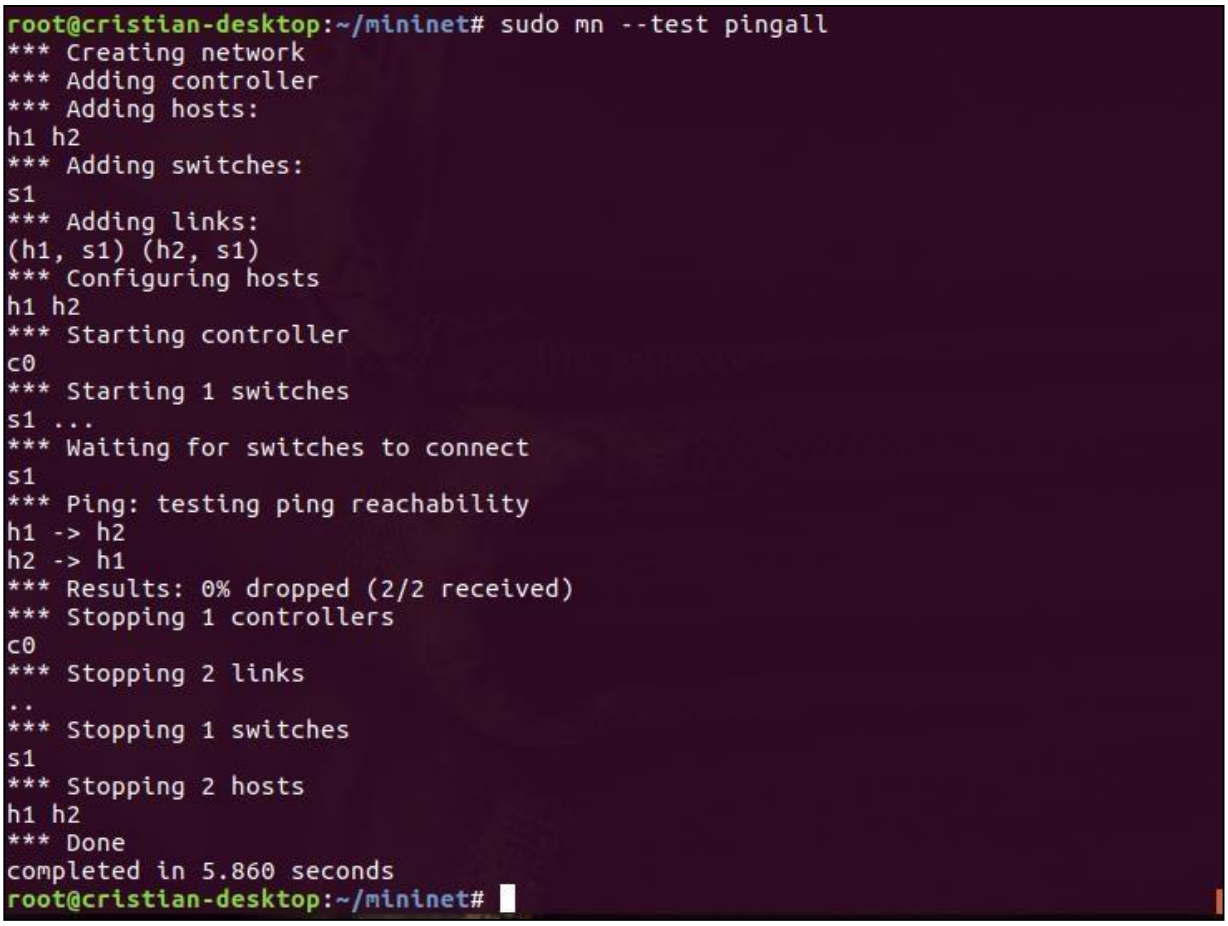

Figura 6.. Instalación de Mininet

Fuente: Autor del proyecto

4.11.2 Instalación RYU controller. Es necesario para esta instalación tener Phyton-pip que se instalará con el comando sudo apt install python-pip. Al terminar puede ejecutar el comando pip install ryu.Otra manera de instalarlo es clonando el repositorio de git y ejecutar su instalador.

~$ git clone git://github.com/osrg/ryu.git

~$ cd ryu

~$ python ./setup.py install

En esta instalación se presentan diversos problemas en la aplicación del proyecto, debido a que la red está asegurada por un proxy que impide el acceso a destinos que no contengan el protocolo https explicito, por lo cual se puede modificar el comando tomando como referencia la dirección ofrecida por https. Puede ser que al momento de ejecutar controlador ocasione problemas debido a que no todas las librerías de Python quedaron instaladas y sea necesario instalarlas manualmente.

4.11.3 Instalación de OpenVswitch. Para esta instalación solo basta con ejecutar el comando sudo apt-get install openvswitch-switch. esta herramienta será utilizada dentro de los controladores y se verá su aplicación más adelante.

Esta herramienta puede instalarse juntamente con Mininet ya que es requerida en su instalación y su propio instalador encuentre la versión que más se ajuste a él.

4.12 Diseño y aplicación de redes

Se establecen una red compartida que consta de 1controlador 4 switch y un host por cada switch, se selecciona uno por cada switch para un método practico ya que estos pueden ser N cantidad de hosts. Se utiliza esta topología para el funcionamiento del laboratorio y para poder utilizar las herramientas de generación estándares de Mininet, a su vez, también se pueden generar topologías personalizadas en caso de ser necesario.

Para iniciar se utilizará el controlador simple_switch_13 de RYU de la siguiente manera:

sudo ./bin/ryu-manager --verbose <dirección del controlador>

Tenemos en cuenta que, al iniciar el controlador en primer lugar, la tabla de direcciones generada será las direcciones por defecto o que tengan el momento los hosts.

luego de esto, se genera la topología con Mininet de la siguiente manera

sudo mn --topo=linear,2 --switch ovsk --mac --controller=remote

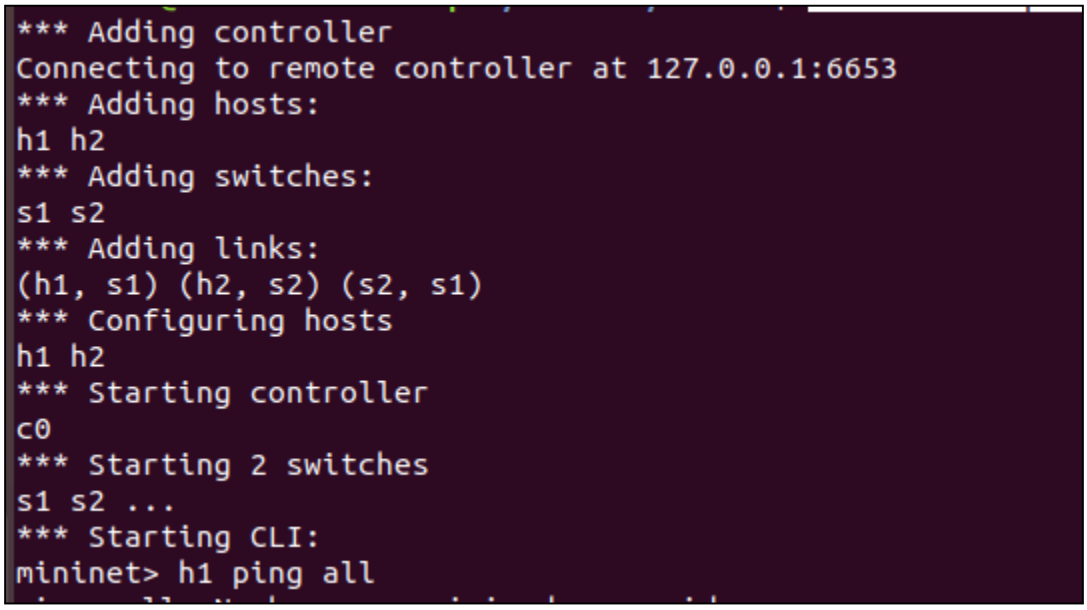

Figura 7.Creación de topología con Mininet,

Fuente: Autor del proyecto

Debe notarse que Mininet conecta el controlador a la dirección 127.0.0.1 y en el puerto 6653 debido a que este se aloja ahí, además, se crean dos hosts, dos switch, y sus correspondientes correcciones.

En primer lugar, se hace un test sobre IPV4, gracias a Mininet se establece un código que realizará los pines correspondientes.

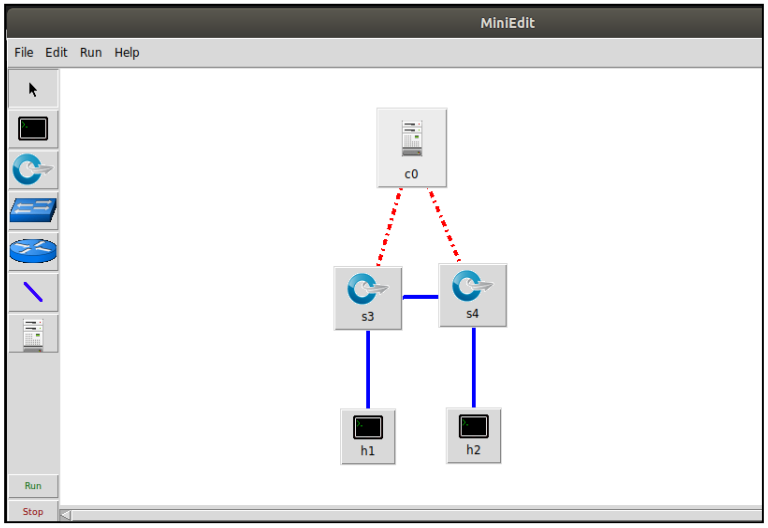

Figura 8. Creación de Topología con miniedit

Fuente: Autor del proyecto

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
```

Figura 9. Test con pingall Ipv6

Fuente: Autor del proyecto

En segundo lugar, se realiza la prueba de conectividad, pero en IPV6:

```
mininet> h1 ping6 -c 3 fe80::200:ff:fe00:2 -I h1-eth0
PING fe80::200:ff:fe00:2(fe80::200:ff:fe00:2) from fe80::200:ff:fe00:1%h1-eth0 h1-eth0: 56 data bytes
64 bytes from fe80::200:ff:fe00:2%h1-eth0: icmp_seq=1 ttl=64 time=0.463 ms
64 bytes from fe80::200:ff:fe00:2%h1-eth0: icmp_seq=2 ttl=64 time=0.101 ms
64 bytes from fe80::200:ff:fe00:2%h1-eth0: icmp_seq=3 ttl=64 time=0.086 ms

--- fe80::200:ff:fe00:2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2035ms
rtt min/avg/max/mdev = 0.086/0.216/0.463/0.175 ms
```

Figura 10. Test ping h1 a h2 IPv6 la primera topología

Fuente: Autor del proyecto

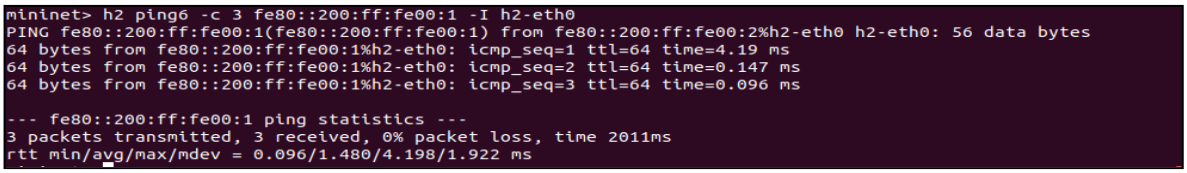

Figura 11. Test ping h2 a h1 IPv6 la primera topología

Fuente: Autor del proyecto

Esta prueba se realiza con base a los comandos establecidos con Mininet de los cuales se describen a continuación:

h2: nombre del host del cual se realizará el ping

ping6: Protocolo que se utilizara

-c 3: Cantidad de ping que se desea hacer

fe80::200:ff:ff:fc00:1 : dirección ipv6 a la cual se dirige

-I h2-eth0: interfaz por la cual se saldrá el paquete.

De igual manera se procede a cambiar el direccionamiento IPV6 para probar la versatilidad del controlador.

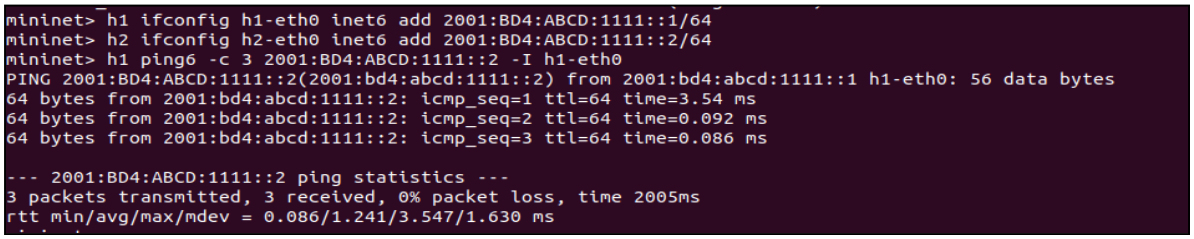

Figura 12. Test ping h1 a h2 IPv6 cambio de dirección

Fuente: Autor del proyecto

Como se puede mostrar en la imagen, se utiliza el comando h1 ifconfig h1-eth0 add 2001:BD4:ABCD:1111::1/64, para añadir la nueva dirección ip, de la misma forma se realiza con el segundo host, y se prueba nuevamente la conectividad entre ellos, dando como resultado que los tres pines se realizaron satisfactoriamente en un tiempo de 2005 ms.

4.12.1 Modelo Propuesto. El modelo propuesto 4 Switch, 4 host, donde los switch están conectados al controlador y este se encargará de administrarlos.

Tabla 5.

Direccionamiento IPv6

HOST.	ENLACE	DIRECCIÓN
H1	Eth-0 inet6	2001:BD4:ABCD:1111::1/64
H2	Eth-0 inet6	2001:BD4:ABCD:1111::2/64
H3	Eth-0 inet6	2001:BD4:ABCD:1111::3/64
H4	Eth-0 inet6	2001:BD4:ABCD:1111::4/64

Fuente: autor

Para la creación de esta red en Mininet se encuentran dos formas que son el modo gráfico y el modo de consola.

Modo gráfico: Este modo se inicializa el Mininet con el siguiente comando **sudo ~/mininet/examples/miniedit.py** de la cual aparecerá la siguiente ventana.

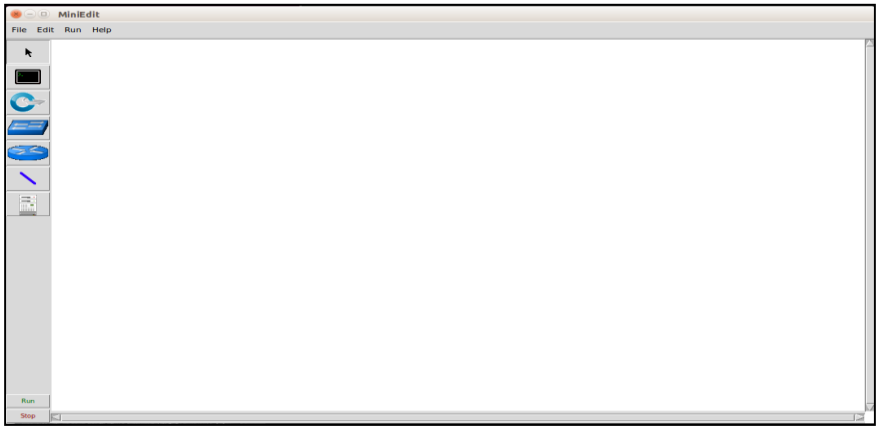

Figura 13. Iniciación de miniedit,

Fuente: Autor del proyecto

De la cual habrá diferentes opciones como lo son:Host, switch virtual, Router, cable, Controlador. Para este caso en el esquema planteado los switch serán reemplazados por switch virtuales para que puedan entender al controlador.

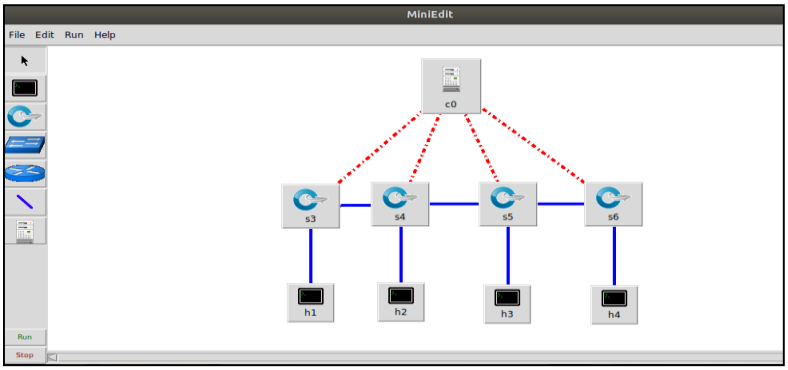

Figura 14. Topología general con miniedit.

Fuente: Autor del proyecto

También se puede generar la topología a través de un código en Python o se pueden establecer con json.

NOTA: Aunque es más fácil generar las redes de forma gráfica, pero algunas funciones no están permitidas y se deberán ejecutar en la consola, como lo son la aplicación del protocolo ipv6, actualmente solo se pueden establecer direcciones Ipv4 de forma gráfica.

Empezaremos creando una topología a través de la herramienta Mininet, ejecutaremos el siguiente comando:

```
# sudo mn --topo=linear,2 --switch ovsk --mac --controller=remote
```

donde:

--topo=linear,2	Equivale al tipo de topología y grado de la misma.
switch ovsk	Hace referencia a el tipo de switch que se va utilizara.
--mac	Habilita las direcciones mac.
--controller=remote	Se establece un controlador externo, solo si es necesario y no se ejecuta en la misma máquina.

Se debe activar el protocolo Ipv6 con el comando **sh sysctl net.ipv6.conf.all.disable_ipv6=1** además al host se debe asignar ip de la forma como se ve en la imagen.

Se procede asignarle a los switch el protocolo OpenFlow 1.3 utilizando la herramienta xterm, de la siguiente manera xterm **(elemento),** se abrirá una nueva ventana donde se le asigna dicho protocolo **ovs-vsctl set Bridge s1 protocols=OpenFlow13**.

Para asignar el controlador se utiliza una nueva ventana y se ejecuta el comando **sudo ./bin/ryu-manager --verbose <dirección del controlador>**el cual iniciara el controlador RYU, se resalta que por defecto se cuenta con varios controladores, en este caso se selecciona el controlador **switch_13,** el cual hará que todos los switch actúen como hub e inunden la red para completar la tabla de direcciones del controlador .

Las siguientes imágenes muestran el comportamiento del controlador a la llegada de solicitudes para conectarse a él.

Figura 15.Iniciación del controlador

Fuente: Autor del proyecto

El controlador acepta las peticiones enviadas responde con un mensaje de saludo “hello”, esto lo hace con todos los switch que se conecten a él, identificando los atributos necesarios, es de tener en cuenta que este controlador trabaja con la versión openflow 1.3

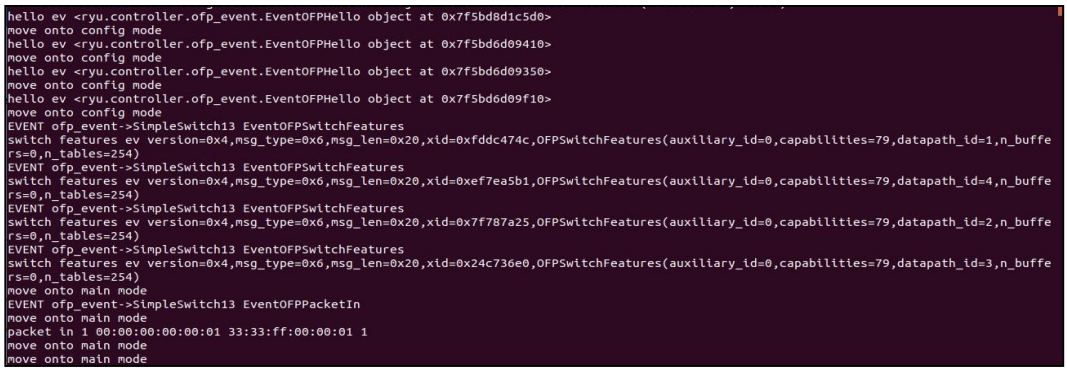

Figura 16. Mensajes “Hello” del controlador

Fuente: Autor del proyecto

Procedemos a realizar las configuraciones en los hosts, añadiendo las direcciones IPV6 y realizamos un ping del host más lejano a partir del host número 1

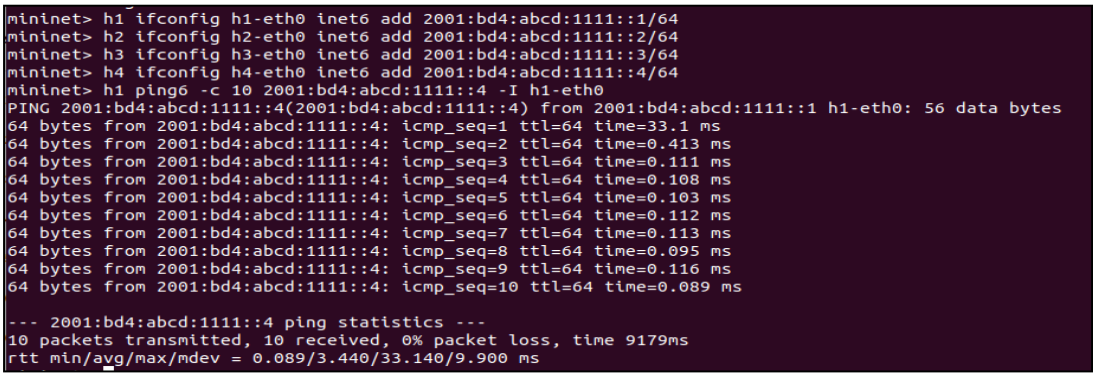

Figura 17.Configuración del entorno virtual

Fuente: Autor del proyecto

Como resultado de esto los diez pines enviados son certeros y llegan a su objetivo, veremos el comportamiento de la red con la herramienta wireshark.

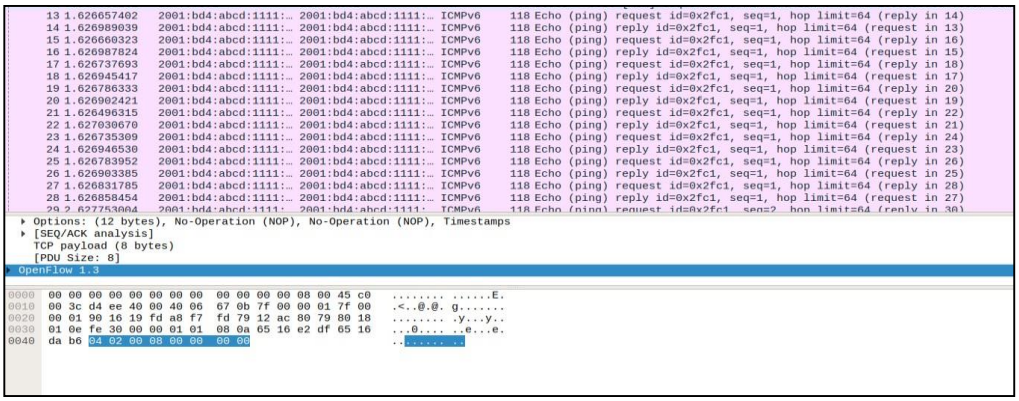

Figura 18. Detección con wireshark,

Fuente: Autor del proyecto

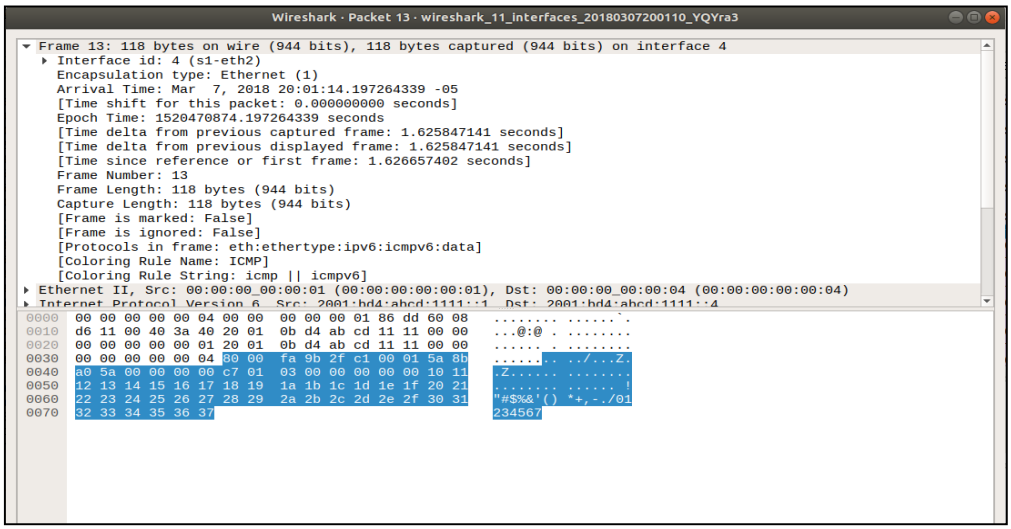

Figura 19. Revisión con wireshark,

Fuente: Autor del proyecto

Verificando conectividad con tracer se utiliza el comando **h1 tracert6 2001:bd4:abcd:1111::4,** donde h1 es el host de donde se va a enviar tracer6 es el comando reservado para ipv6 y se anexa la dirección ipv6 del receptor, esta prueba se aplica para todos los host, se resalta que el controlador es aquel que está actuando como enrutador entre los switch.

```
mininet> h1 tracert6  2001:bd4:abcd:1111::4
traceroute a 2001:bd4:abcd:1111::4 (2001:bd4:abcd:1111::4) de 2001:bd4:abcd:1111::1, 30 saltos máx, Paquetes de 60 bytes
 1  2001:bd4:abcd:1111::4 (2001:bd4:abcd:1111::4)  0.477 ms  0.001 ms  0.001 ms
mininet> h2 tracert6  2001:bd4:abcd:1111::4
traceroute a 2001:bd4:abcd:1111::4 (2001:bd4:abcd:1111::4) de 2001:bd4:abcd:1111::2, 30 saltos máx, Paquetes de 60 bytes
 1  2001:bd4:abcd:1111::4 (2001:bd4:abcd:1111::4)  35.448 ms  0.124 ms  0.071 ms
mininet> h3 tracert6  2001:bd4:abcd:1111::4
traceroute a 2001:bd4:abcd:1111::4 (2001:bd4:abcd:1111::4) de 2001:bd4:abcd:1111::3, 30 saltos máx, Paquetes de 60 bytes
 1  2001:bd4:abcd:1111::4 (2001:bd4:abcd:1111::4)  17.887 ms  0.886 ms  0.281 ms
mininet> h4 tracert6  2001:bd4:abcd:1111::4
traceroute a 2001:bd4:abcd:1111::4 (2001:bd4:abcd:1111::4) de 2001:bd4:abcd:1111::4, 30 saltos máx, Paquetes de 60 bytes
 1  cristian-desktop (2001:bd4:abcd:1111::4)  0.002 ms  0.001 ms  0.001 ms
```

Figura 20. Comando Tracert

Fuente: Autor del proyecto

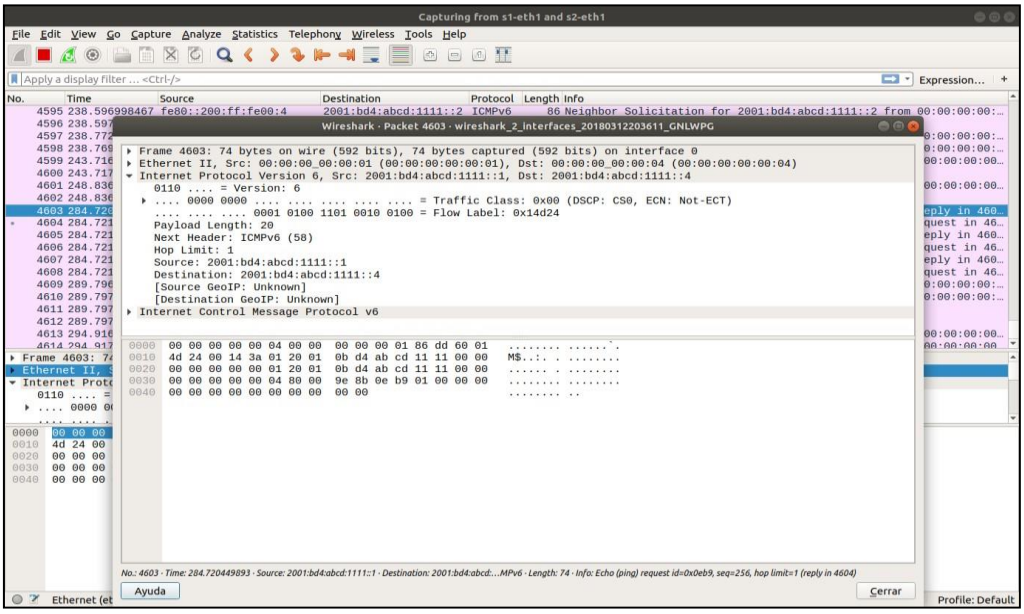

Figura 21. Comando Tracer wireshark

Fuente: Autor del proyecto

Se verifica la comunicación mediante wireshark, se tiene en cuenta que solo se analiza la trama del switch1 y el switch4

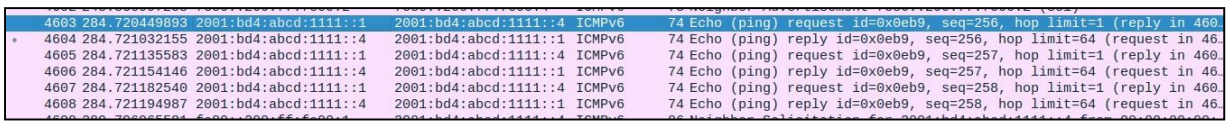

Figura 22. Trama Switch 1 a Switch 4

Fuente: Autor del proyecto

Se utiliza el comando iperf para crear flujos de datos UDP y medir el rendimiento de la red creada, se utiliza UDP ya que permite especificar el tamaño de los datagramas y proporciona resultado del rendimiento y paquetes perdidos, además, nos permite hacer funcionar los hosts como cliente y servidor para medir rendimiento entre dichos extremos. Este comando es aplicado de la siguiente manera:

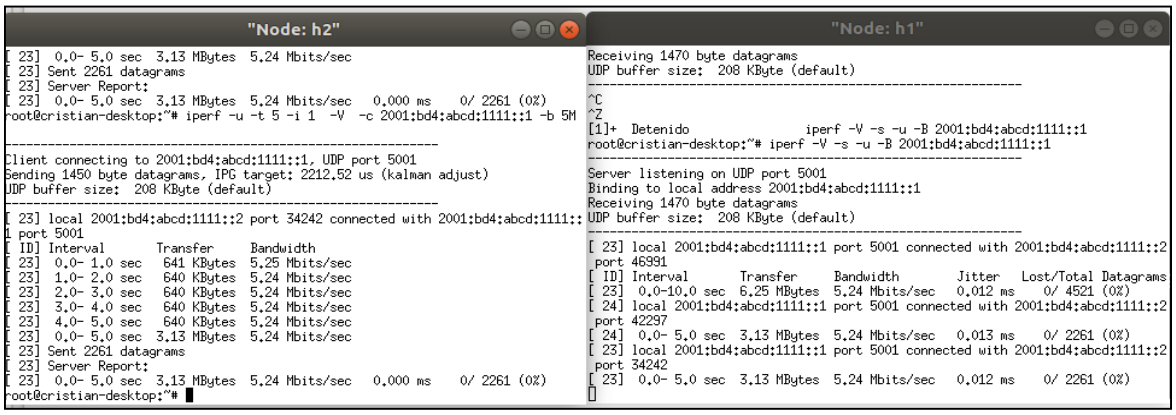

Figura 23. Comando Iperf,

Fuente: Autor del proyecto

El servidor de iperf se instala en el host1 con el siguiente comando **iperf -V -s -u -B 2001:bd4:abcd:1111::1** donde -V Habilita la recepción de IPv6 configurando el dominio y el socket en AF_INET6 (Puede recibir tanto en IPv4 como en IPv6), -s ejecuta el host como servidor, -u utiliza solamente UDP , -B enlazar a <host>, una interfaz o dirección de multidifusión.

Para el cliente, en este caso se utiliza el host2, utilizando los siguientes comandos **iperf -u -t 5 -i 1 - V -c 2001:bd4:abcd:1111::1,** al igual que en el caso anterior -V se utiliza para activar el ipv6, -u para enviar solamente por UDP, -t tiempo en segundos para escuchar nuevas conexiones

y para recibir tráfico (predeterminado no configurado), -i segundos entre informes periódicos de ancho de banda, - c ejecutar en modo cliente, conectando a <host>.

Como se puede apreciar en la imagen, esta prueba nos arroja la siguiente información, intervalo, transferencia y banda ancha, además de mostrarnos la cantidad de datagramas enviados.

Tabla 6.

Tiempos de respuesta OpenFlow

Número de ping	Tiempo (ms)
1	33.1
2	0.413
3	0.111
4	0.108
5	0.103
6	0.112
7	0.113
8	0.095
9	0.115
10	0.089

Fuente: Autor del proyecto

Tabla 7

Intervalos y transferencia

Intervalo (seg)	Transferencia (KBytes)
0.0 - 1.0	641
1.0 - 2.0	640
2.0 - 3.0	640
3.0 - 4.0	640
4.0 - 5.0	640

Fuente: Autor del proyecto

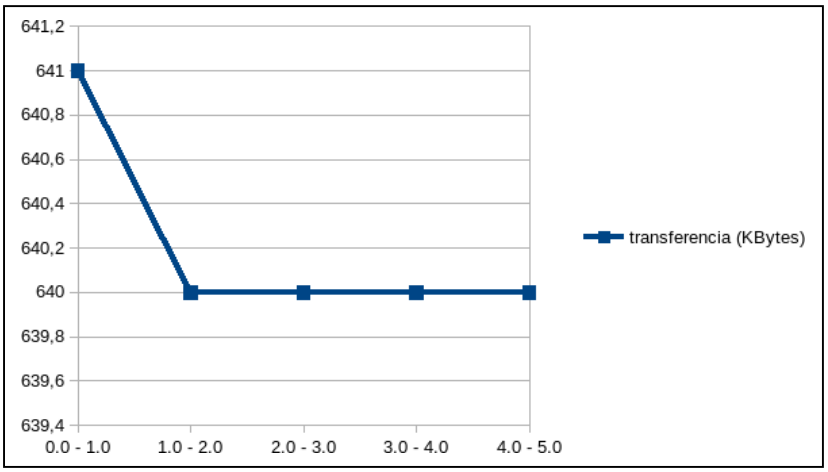

Figura 24. Gráfica de transferencia de datos del ipef en Kbytes

Fuente: Autor del proyecto

Con los datos obtenidos del iperf se establece que la transferencia es estable prácticamente invariable teniendo en cuenta que el primer dato tiene una variación de 641 Kbytes.

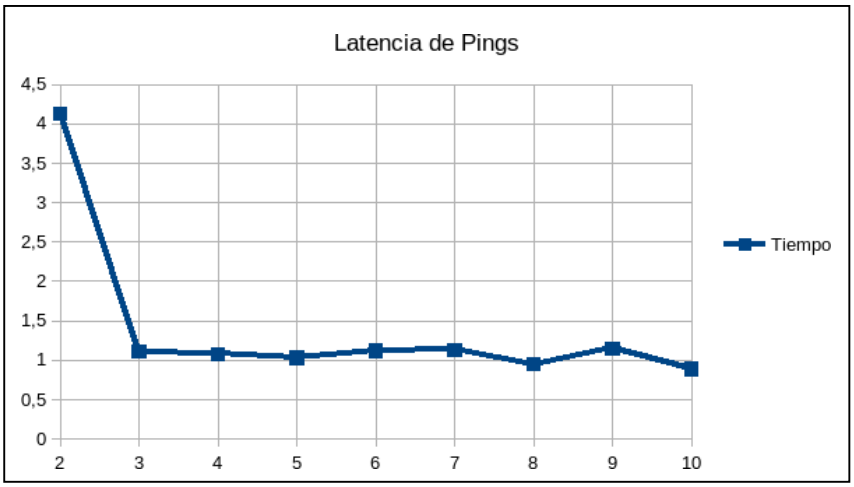

Figura 25. Gráfica de latencia de Pings en intervalo de tiempo

Fuente: Autor del proyecto

Depreciando el primer ping de la tabla de datos la conexión es estable, en promedio es de 1.3988 ms por cada ping enviado, una respuesta rápida para las conexiones establecidas.

Para cada una de las pruebas realizadas se obtiene un 100% de entrega de los pines efectuados con una respuesta rápida lo que evidencia la efectividad del controlador a la hora de establecer conexiones entre host, a su vez puede comunicar con diferentes protocolos proveniente de Openflow 1.3.

Se realiza una conexión similar a la propuesta anteriormente, pero con base a la red clásica y donde se realizan los pings correspondientes por cada host, en este caso host1 y host2 obteniendo los siguientes resultados.

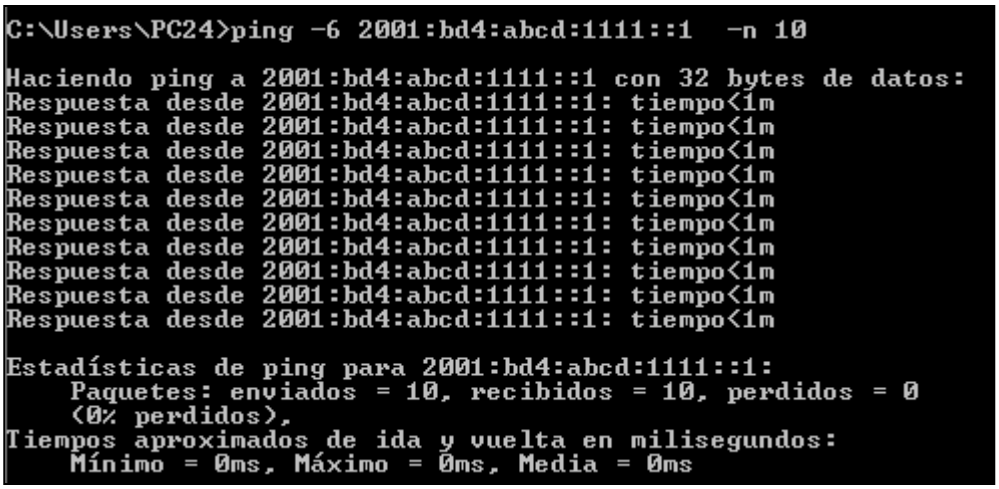

Figura 26. Ping red tradicional

Fuente: Autor del proyecto

Como se puede observar los pines son 100% efectivos y se pueden analizar en wireshark de donde se extrae el tiempo de ejecución para la comparativa.

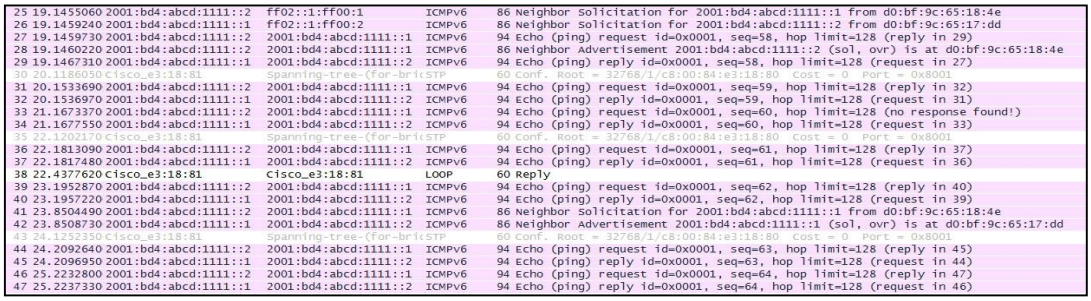

Figura 27. . Wireshark red Tradicional.

Fuente: Autor del proyecto

Tabla 8.

Tiempos de respuesta red tradicional, Fuente Autor del proyecto

Ping	Tiempo de respuesta (ms)
1	0.709
2	0.418
3	0.439
4	0.435
5	0.424
6	0.431
7	0.453
8	0.50
9	0.51
10	0.34

Como se puede observar los datos son bastantes similares a los obtenidos con el tipo de red que contiene el protocolo OpenFlow 1.3

Firewall

Para comprobar la versatilidad de las redes definidas por software se utiliza un nuevo controlador llamado rest_firewall, como su nombre se indica con este controlador se programara un firewall para bloquear los pines que se realizan en los host, al igual que en el ejercicio anterior se lanza el controlador y luego se inicializa la topología, pero en este caso se usa el comando **sudo mn --topo=single,3 --mac --controller remote -x**, este comando se diferencia del anterior en el tipo de topología utilizado, el cual se generará con un router y 3 host(vea la imagen x.x), a su vez se agrega el comando -x, el cual servirá para abrir todos los dispositivos de forma remota

con xterm y poder configurarlos, a los host se les asignaron las mismas direcciones que en el ejercicio pasado de la misma manera y se activa el protocolo OpenFlow 1.3, luego de ellos nos dirigimos al controlador a configurar el firewall.

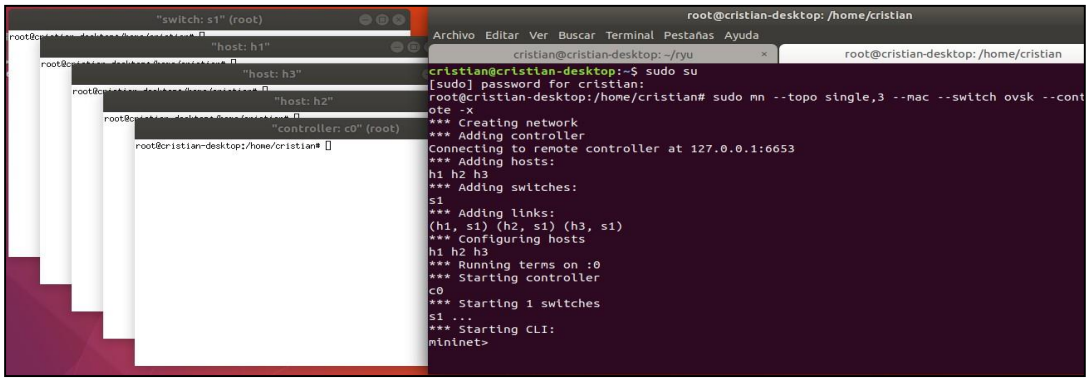

Figura 28. Creación de topología Mininet con xterm

Fuente: Autor del proyecto

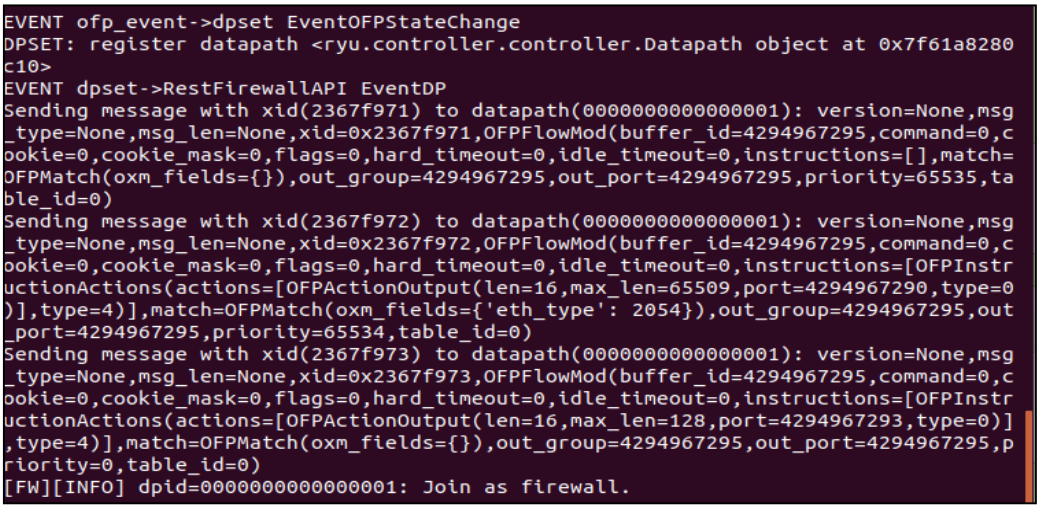

Figura 29.Iniciación del controlador Ryu Firewall

Fuente: Autor del proyecto

Detección del dispositivo en el controlador

Para las configuraciones del Firewall se utiliza curl y se envían las reglas de la siguiente manera, **curl -X PUThttps://127.0.0.1:8080/firewall/module/enable/0000000000000001** con este comanda se activa el firewall devolviendo un mensaje de activación, luego de ello procedemos a verificar el estado del mismo de la siguiente manera **curl https://127.0.0.1:8080/firewall/module/status.**

Figura 30.Iniciación de firewall mediante curl,
Fuente: Autor del proyecto

Estando todo en perfecto orden iniciamos agregando las reglas:

curl -X POST -d '{"ipv6_src" : "2001:bd4:abcd:1111::1" , "ipv6_dst" :
"2001:bd4:abcd:1111::2", "nw_proto" : "ICMPv6" }'
http://127.0.0.1:8080/firewall/rules/0000000000000001

Como se puede apreciar se utiliza nuevamente el comando curl enviando una petición POST con los datos de ip de origen, ip destino y protocolo, a su vez a donde va dirigido y qué elemento de la red va a portar la regla, se realiza la misma acción los host que se requieran aplicar las reglas, Para terminar y guardar las reglas se utiliza el comando, curl http://127.0.0.1:8080/firewall/rules/0000000000000001/all el cual dará respuestas de las reglas que se han guardado, cabe resaltar que se puede agregar en la configuración un action que puede

llevar como comandos = ALLOW o DENY, el cual permitirá o denegara el acceso de la ip origen a la ip destino(Figura 29), se puede comprobar las configuración realizando los pings correspondientes a cada host. teniendo en cuenta que se denegó el ping a ciertos host, por defecto todos están en estado DENY. Se resalta que al agregar la dirección que permita la conexión entre los hosts también se debe agregar la dirección multicast, si esta no está agregada el ping igualmente será bloqueado (Figura 30).

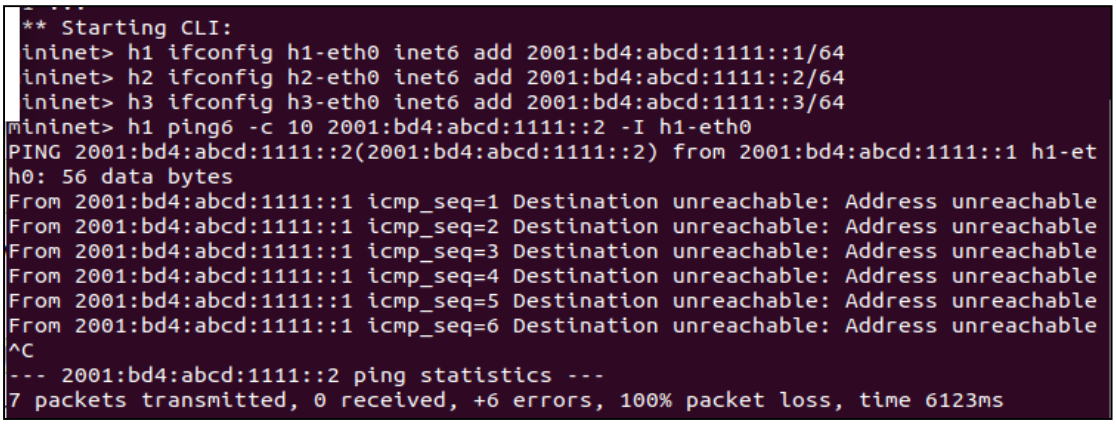

Figura 31. Ping denegado por firewall

Fuente: Autor del proyecto.

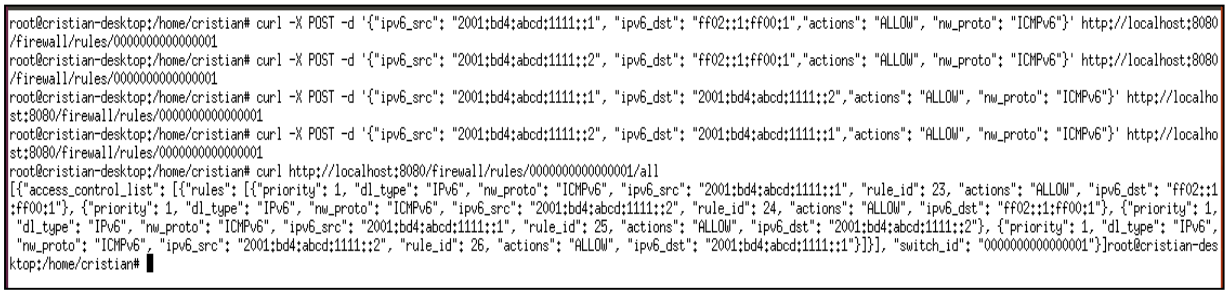

Figura 32. Adición de reglas al firewall

Fuente: Autor del proyecto

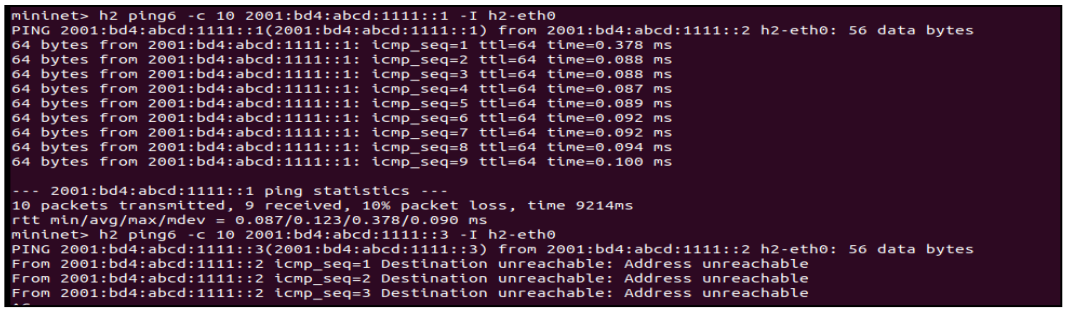

Figura 33. Ping recibidos y ping denegado Firewall

Fuente: Autor del proyecto

Tabla 9

Tiempos de Respuesta, Openflow Firewall

Número de ping	Tiempo (ms)
1	0.378
2	0.088
3	0.088
4	0.087
5	0.089
6	0.092
7	0.092
8	0.094
9	0.100

Fuente: Autor del proyecto

Firewall utilizando vlan

El procedimiento a ejecutar es muy parecido al anterior solo que en vez de denegar acceso de host directamente se negaran a las vlan correspondiente para evitar su comunicación y que al

host los enlazaremos a las vlans correspondientes, para dicho procedimiento asignaremos al host las ip y las vlans.

```
mininet> h1 ip add add 2001:bd4:abcd:1111::1/64 dev h1-eth0.1
mininet> h1 ip link set dev h1-eth0.1 up
mininet> h2 ip link add link h2-eth0 name h2-eth0.1 type vlan id 1
mininet> h2 ip addr add 2001:bd4:abcd:1111::2 dev h2-eth0.1
mininet> h2 ip link set dev h2-eth0.1 up
mininet> h3 ip link add link h3-eth0 name h3-eth0.2 type vlan id 2
mininet> h3 ip addr add 2001:bd4:abcd:1111::3/64 dev h3-eth0.2
mininet> h3 ip link set dev h3-eth0.2 up
mininet> h4 ip link add link h4-eth0 name h4-eth0.2 type vlan id 2
mininet> h4 ip addr add 2001:bd4:abcd:1111::4/64 dev h4-eth0.2
mininet> h4 ip link set dev h4-eth0.2 up
mininet>
```

Figura 34. Creación de Vlans,

Fuente: Autor del proyecto

De esta forma ya habremos asignado el host 1 y el host 2 a la vlan 10, y por su parte el host 3 y host 4 a la vlan2 con su correspondiente subinterfaz, luego de ello procedemos a darle las reglas al controlador para que estas se puedan comunicar dónde quedaría de la siguiente manera.

```
root@cristian-desktop:/home/cristian# curl http://localhost:8080/firewall/rules/0000000000000001/all
[{"access_control_list": [{"rules": [{"priority": 1, "dl_type": "IPv6", "nw_proto": "ICMPv6", "ipv6_src": "2001:bd4:abcd:1111::3", "dl_vlan": "2", "rule_id":
1, "actions": "ALLOW"}, {"priority": 1, "dl_type": "IPv6", "nw_proto": "ICMPv6", "ipv6_src": "2001:bd4:abcd:1111::4", "dl_vlan": "2", "rule_id": 2, "actions":
 "ALLOW"}], "vlan_id": "2"}], "switch_id": "0000000000000001"}]root@cristian-desktop:/home/cristian#
```

Figura 35. Adición de reglas Firewal vlan,

Fuente: Autor del proyecto

Realizamos los pines de conectividad para comprobar que los hosts de la vlan 10 no puedan acceder al host de la vlan 2, y que por su parte la vlan 2 puedan acceder a su host.

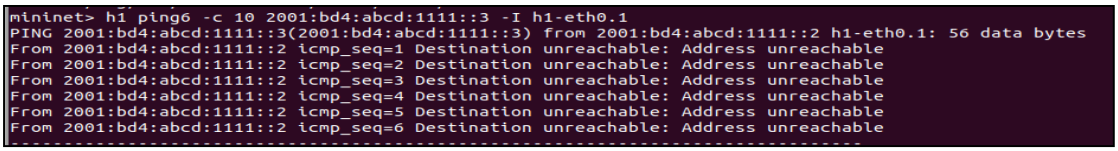

Figura 36. Ping denegado Firewall vlan.

Fuente: Autor del proyecto

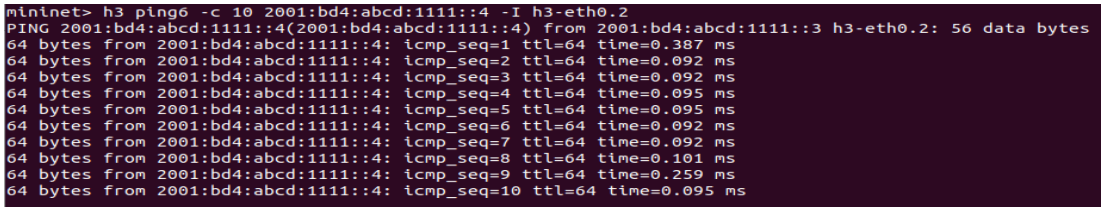

Figura 37. Ping acertado Firewall vlan,

Fuente: Autor del proyecto

Tabla 10

Tiempos de respuesta Firewall Openflow. Fuente: Autor del Proyecto

Número de pin	Tiempo (ms)
1	0.387
2	0.092
3	092
4	0.095
5	0.095
6	0.092
7	0.092
8	0.101
9	0.259
10	0.095

Fuente: Autor del proyecto

Se utiliza el esquema propuesto para probar la efectividad de controlador con más de un elemento de red, se realizan las operaciones correspondientes, como lo es la creación de la red a través de Mininet, la aplicación de dirección y el lanzamiento del controlador. En el controlador se activa el firewall mediante curl pero en este caso como se va a aplicar las reglas a dos dispositivo se debe realizar para cada uno pero en el mismo controlador.

Tabla 11

Acciones Firewall host 1 . Fuente Autor del proyecto

IP de envío	IP de destino	Dispositivo	Acción
2001:bd4:abcd:1111::1	2001:bd4:abcd:1111::2	S1	ALLOW
2001:bd4:abcd:1111::1	2001:bd4:abcd:1111::2	S2	ALLOW
2001:bd4:abcd:1111::1	ff02::1:ff00:2	S2	ALLOW
2001:bd4:abcd:1111::1	ff02::1:ff00:2	S1	ALLOW

Fuente: Autor del proyecto

De esta forma el host1 sería el único que accedería al host2, pero el host2 no podría acceder al host 1.

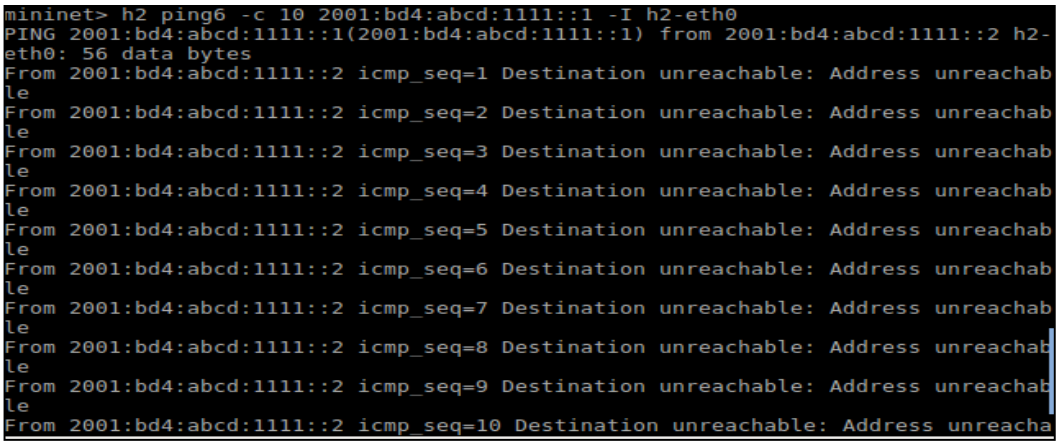

Figura 38. Ping denegado host 2 a host 1 Firewall Vlan,

Fuente: Autor del proyecto

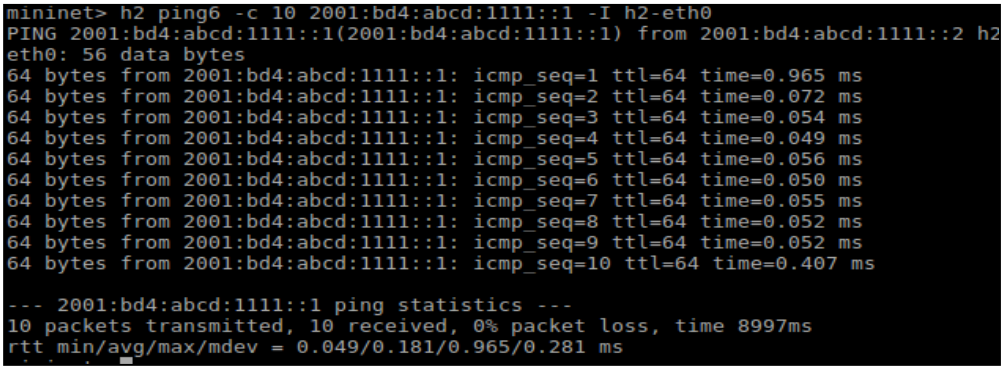

Figura 39. Ping recibido host 1 a host 2 Firewall Vlan,

Fuente: Autor del proyecto

Para que el host2 pueda acceder al host1, debemos agregarle nuevas reglas que permitan dicha comunicación de la siguiente manera

Tabla 12

Acciones Firewall host 2. Fuente Autor del proyecto

IP de envío	IP de destino	Dispositivo	Acción
2001:bd4:abcd:1111::2	2001:bd4:abcd:1111::1	S1	ALLOW
2001:bd4:abcd:1111::2	2001:bd4:abcd:1111::1	S2	ALLOW
2001:bd4:abcd:1111::2	ff02::1:ff00:1	S2	ALLOW
2001:bd4:abcd:1111::2	ff02::1:ff00:1	S1	ALLOW

Fuente: Autor del proyecto

Al establecer estas reglas los pings entre estos hosts deben ser efectivos.

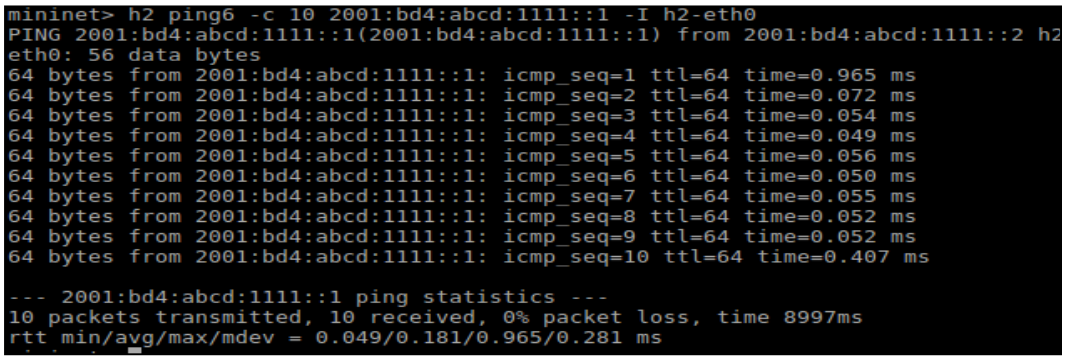

Figura 40. Ping recibido host 2 a host 1 Firewall Vlan,

Fuente: Autor del proyecto

Tenemos en cuenta que las configuraciones no se hicieron entrando al switch1 y luego al switch2, estas reglas se le asignaron al controlador directamente, indicando a cuál de ellos

afectaba y que debería hacer con los paquetes provenientes de una dirección centralizando la configuración de la red.

Figura 41. Reglas establecidas en el controlador Firewall Vlan,

Fuente: Autor del proyecto

Para mayor depuración del administrador de la red, el controlador arroja información de lo que está ocurriendo, que paquete bloquea y que paquete llega a su destino y en donde ocurrió el evento, a su vez da información de las cabeceras utilizadas, como lo son la versión del protocolo, el tipo de paquete, tamaño de paquete entre otras.

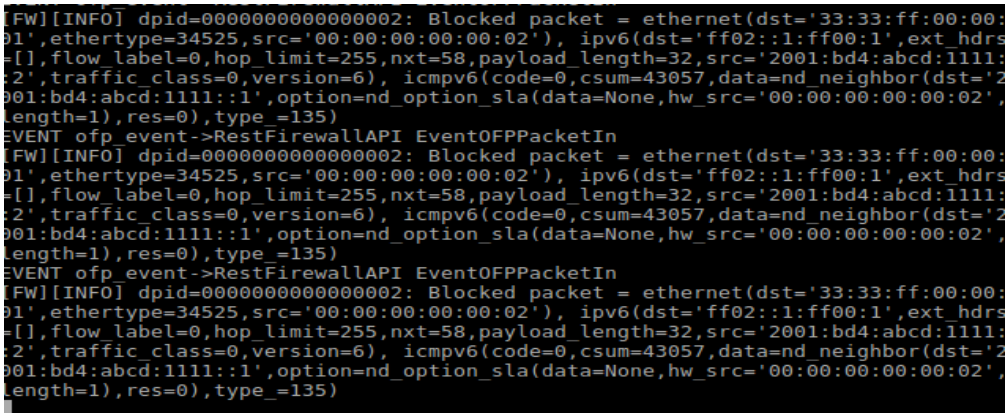

Figura 42. Seguimiento del controlador a la red.

Fuente: Autor del proyecto

4.13 Diagnóstico final

Con los datos obtenidos mediante el controlador RYU para la migración de la red estándar a una con la aplicación del protocolo OpenFlow, hay diferentes requerimientos que se deben tener en cuenta: los dispositivos que soporten el protocolo OpenFlow, se debe desarrollar un controlador que se adapte a las necesidades de la empresa(debido que los desarrollados actualmente se ocupan de ciertos campos y no cumplen con todo lo requerido), aunque algunos de ellos como se demostró anteriormente con simple_switch_13 y rest_firewall cubren muchas de ellas, se optó por utilizar la versión 1.3 de OpenFlow en el controlador ya que los protocolos utilizados están habilitados para esta versión y no se hace necesario utilizar las versiones 1.4 y 1.5. (*Ver Apéndice 2: Diferentes opciones correspondientes a ipv6 que maneja la versión 1.3*).

Tabla 13.

Ventajas y desventajas .

Ventajas	Desventajas
Centralización de configuración de red	Seguridad
Reducción de costos	Gastos de nuevos equipos
Fiabilidad y menor tiempo de inactividad	Confiabilidad y desempeño frente a redes actuales
Adaptable y expandible	Características y funciones inconsistentes

Fuente: Autor del proyecto

Debemos tener en cuenta que las ventajas y desventajas pueden variar dependiendo del punto de vista que se mire. Por ejemplo, Seguridad, aunque en este caso se establece en desventaja, ya que centralizamos toda la configuración en un punto y para un atacante es más fácil acceder a un solo punto, pero la seguridad seria ventaja, porque la red es programable y podemos encriptar la misma y sería muy difícil para un atacante descifrar nuestra configuración.

Para la migración total de la red a una tecnología SDN se deben tener en cuenta diferentes aspectos, tales como:

¿Los equipos de red son los suficientemente reciente o compatibles con una red SDN?

¿Es necesario una actualización?

¿Tiene un problema de red que justifique una actualización fuera de tiempo?

¿Puede su proveedor brindarle una arquitectura de red que satisfaga sus requerimientos específicos?

Entre estas y otras preguntas se plantea el concepto de implementación de una red SDN. Dado que dicha migración genera gastos y requerimientos tecnológicos no solo por parte de la

empresa sino también de los proveedores, puede que la implementación de esta tecnología no se pueda realizar o no sea necesaria todavía su implementación.

Una integración total entre estos controladores satisfacerá muchas de las necesidades de cualquier empresa, aportando tanto como velocidad en sus redes, una centralización en la configuración de red y seguridad en sus datos. se destaca que es una herramienta de fácil manejo y con una amplia documentación que conlleva a un crecimiento rápido en el conocimiento de la misma y en el manejo de sus redes.

Ryu controller aparte de todos los beneficios presentados anteriormente presenta compatibilidad con diferentes herramientas que gran ayuda para los administradores de redes, como lo es ryu VRRP el cual hace testeo a la red dándonos información de lo que ocurre en nuestro esquema, a su vez cuenta con snort el cual es un famoso detector de intrusos de carácter libre, puede actuar como sniffer de paquete directo como tcpdump y puede depurar el tráfico de la red, además, se tiene una amplia documentación y un repositorio en git donde se evidencian ejemplos y explicaciones realizados por comunidades interesadas, lo que puede facilitar el desarrollo de nuevos controladores y aplicarlos al modelo de negocio que se plantea.

Se recomienda el desarrollo de aplicaciones SDN independientemente del controlador que se escoja ya que los actuales se realizan para satisfacer necesidades netamente de aprendizaje. Cabe resaltar que las SDN se basan primeramente en la implementación en laboratorios de forma virtual como se presentó en este proyecto, y luego de ellos se puede implementar de forma física e instanciarlo en la nube.

4.13.1 Posibles errores encontrados en la instalación. Debido a las recientes actualizaciones del sistema operativo algunos controladores no funcionan de forma correcta como es el caso de HP VAN SDN controller (*Ver Apéndice 1: Instalación de otros controladores que son de utilidad*), este controlador solo funciona en las versiones de Ubuntu 12 hasta la versión 14 con algunos problemas de instalación, en la versión más recientes dicho controlador queda obsoleto, aunque se ha actualizado a una versión para sistemas operativo más actualizados y es un controlador más potente este deja de ser de carácter libre.

En Mininet debido al tipo de instalación que se efectúe se pueden obtener problemas con permisos de ejecución en especial al invocar al programa externo xterm, si el error es producido, se debe aplicar el siguiente comando: xhost +, el cual permitirá a cualquier usuario ejecutar dicho programa.

Así mismo, en algunos casos, el software Mininet no encuentra el controlador, esto puede ocurrir debido a que hay muchos controladores en diferentes puertos y este no entienda a cuál debe conectarse, para evitar este tipo de inconvenientes cada vez que termine de utilizar un controlador termine la ejecución del proceso de manera correcta.

Capítulo 5. Conclusiones

Teniendo en cuenta las necesidades basadas en este proyecto y los objetivos planteados se optó por seleccionar el controlador RYU, ya que con sus características otorga una amplia gama de opciones a la hora de desarrollar e implementar una red definida por software, pero esto no quiere decir que otros controladores de carácter libre no estén a un mismo nivel, algunos poseen características similares a este y también se pueden adaptar a requerimientos de las empresas o instituciones, por ejemplo NOX130FLIB, Trema y Opendayligh, donde se pueden también utilizar como frameworks y realizar controladores propios.

El ambiente virtual realizado con la herramienta Mininet favorece a la aplicación y prueba de los controladores, dando una simulación exacta de lo que ocurre en un entorno real, permitiendo el manejo del protocolo Ipv6 y el protocolo OpenFlow dando una experiencia agradable y practica de esta tecnología, a su vez, se pueden modelar diferentes tipos de redes, desde unas muy simples a otra más complejas, ya sea desde su parte gráfica, consola o programándolas con el lenguaje Python, aunque para controladores que se encuentran alojados de manera externa no se puede hacer conexión con ellos a través del protocolo IPv6.

Las SDN son una gran alternativa con una compatibilidad alta con el protocolo IPv6, a su vez cuenta con diferentes versiones que cada vez se expanden y profundizan más en dicho protocolo haciendo así una propuesta viable para la migración de redes, no obstante, la forma en cómo se elabora o implemente el controlador depende de las necesidad y de cómo se concibe la red en la organización, por lo tanto, es un reto para el desarrollador del mismo, aunque esto da

una gran ventaja, ya que este conoce como fluctúan los cambios de red y podrá respaldar rápidamente cualquier falla que se presente o cualquier cambio que sufra el modelo de red. Se comprueba la separación del plano de datos con el plano de control, y la centralización de la configuración de la red, permitiendo un mayor control de la misma, En comparación con la red tradicional las velocidades se mantienen, es decir que al migrar a esta tecnología los tiempos de respuesta se van a mantener y no se perderá calidad en la misma. Aunque hay que tener en cuenta los diferentes aspectos que conlleva una migración de red, la red definida por software se convierte en una solución factible o digna de evaluar cuando se lleve dicho proceso.

Capítulo 6. Recomendaciones

Durante el desarrollo del proyecto del proyecto se realizaron pruebas con diferentes soluciones, en las cuales se presentaron diversas dificultades a la hora de realizar la implementación.

Debido a las actualizaciones del sistema operativo se ocasionaron fallas en la instalación de las herramientas como Mininet y RYU controller provocando que los repositorios de movieran y no se encontraran plugins necesarios para su ejecución.

Para Mininet el tener más de un controlador en ejecución provocaba conflicto interno, ocacionando las redes virtualizadas no se crearán y el programa se cerraba de golpe, en alguno de los casos desintalaba parte de dicho programa.

Por ultimo las redes virtualizadas ocasionaban problemas cuando se trabajaban de forma gráfica (Miniedit) debido a que no se reconoce el protocolo IPv6 en ellas.

En consecuencia, a las razones anteriores se realizan estas sugerencias para los lectores:

Antes de instalar las herramientas verificar la compatibilidad de las herramientas con la versión actual de sistema operativo, en caso contrario instalar las recomendadas e instalar todos los repositorios necesarios para dichas herramientas

Cada vez que se realizan pruebas con diferentes controladores, debe terminar la ejecución de ellos debidamente antes de iniciar otro de ellos o instanciar debidamente el puerto donde se va a ejecutar.

Trabajar las redes virtualizadas por la interfaz de comandos y para las redes personalizadas utilizar phyton preferiblemente.

Referencias

Ali Al-Shabibi. (2013, Febrero 19). *OpenFlow stanford.* Retrieved from https://openflow.stanford.edu/display/ONL/POX+Wiki

Big Switch Network. (2014). *bigswitch.* Retrieved from https://www.bigswitch.com/blog/2014/12/22/2014-what-a-difference-a-year-makes

CISCO. (2013, Marzo 1). *CISCO.* Retrieved from https://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-59/161-sdn.html

Citrix Systems, Inc. (2014). *Mobilize Your Business with Secure App and Data Delivery - Citrix.* Retrieved from https://www.citrix.com/content/dam/citrix/en_us/documents/oth/sdn-101-an-introduction-to-software-defined-networking-es.pdf

Community, R. S. (2017). *github.* Retrieved from https://osrg.github.io/ryu/

David Bombal, J. D. (2018, Mayo 17). *GNS3.* Retrieved from https://docs.gns3.com/1PvtRW5eAb8RJZ11maEYD9_aLY8kkdhgaMB0wPCz8a38/index.html

Erickso, D. (2012, Junio 22). *Home Beacon Confluence*. Retrieved from https://openflow.stanford.edu/display/Beacon/Home

Erickso, D. (2013, Febrero 14). *Home Beacon Confluence*. Retrieved from https://openflow.stanford.edu/display/Beacon/Home

Feamster, N. (2014, Abril 2). *acm.* Retrieved from https://dl.acm.org/citation.cfm?id=2602219

Hewlett-Packard Development Company, L.P. (2013, Septiembre). *hpe.* Retrieved from http://h17007.www1.hpe.com/docs/networking/solutions/sdn/4AA4-8807ENW.PDF

Huawei Technologies Co. (2018). *huawei.* Retrieved from http://e.huawei.com/in/marketing-material/onLineView?MaterialID=%7B3C52ACD5-0135-4FC6-9880-7996D7F31342%7D

Ministerio de tecnologías de la información y las comunicaciones. (2009, julio 30). *mintic.* Retrieved from https://mintic.gov.co/portal/604/articles-8580_PDF_Ley_1341.pdf

Ominike, A. (2016, Diciembre). *researchgate.* Retrieved from https://www.researchgate.net/publication/311479628_Introduction_to_Software_Defined_Networks_SDN

Open Network Fundation. (2012, Abril 13). Retrieved from https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf

Open Networking. (2018). *opennetworking.* Retrieved from https://www.opennetworking.org/software-defined-standards/overview/

Open Networking Foundation. (2009, Diciembre 31). Retrieved from Open Networking: https://www.opennetworking.org/wp-content/uploads/2013/04/openflow-spec-v1.0.0.pdf

Open Networking Foundation. (2018). *Open Datapath - Open Networking Foundation*. Retrieved from https://www.opennetworking.org/technical-communities/areas/specification/open-datapath/

Open Networking Foundation. (2018). *Open Networking Foundation*. Retrieved from https://www.opennetworking.org/sdn-definition/

OpenDaylight Project a Series of LF Projects, LLC. (2018). *OpenDaylight.* Retrieved from https://www.opendaylight.org/about/news/blogs/opendaylight-releases-oxygen-with-new-p4-and-container-support

Popoviciu, C. (2013, Diciembre). *techtarget.* Retrieved from https://searchsdn.techtarget.com/feature/IPv6-SDN-When-worlds-collide-in-a-good-way

REPUBLICA DE COLOMBIA – GOBIERNO NACIONAL . (2004, Enero 5). Retrieved from http://www.suin-juriscol.gov.co/clp/contenidos.dll/Leyes/1669934?fn=document-frame.htm$f=templates$3.0

Rouse, M. (2010, Julio). *techtarget.* Retrieved from https://searchservervirtualization.techtarget.com/definition/virtual-switch

Rouse, M. (2012, Noviembre). *What is SDN controller (software-defined networking controller)*. Retrieved from https://searchsdn.techtarget.com/definition/SDN-controller-software-defined-networking-controller

Rouse, M. (2013, Marzo). *SearchSDN*. Retrieved from http://searchsdn.techtarget.com/definition/POX

Rouse, M. (2016, Octubre). *SearchNetworking.* Retrieved from https://searchnetworking.techtarget.com/definition/router

Rouse, M. (2017, Julio). *Search Networking.* Retrieved from https://searchtelecom.techtarget.com/definition/switch

Rouse, M. (2018, Enero). *techtarget.* Retrieved from https://whatis.techtarget.com/definition/server

Secretaría Jurídica Distrital de la Alcaldía Mayor de Bogotá D.C. (2009, julio 30). Retrieved from Alcaldia de Bogotá: http://www.alcaldiabogota.gov.co/sisjur/normas/Norma1.jsp?i=36913

The Open Networking Foundation. (2015, Marzo 26). Retrieved from Open Networking: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf

Trema. (2011, Noviembre 4). *github.* Retrieved from http://trema.github.io/trema/

Yanhe Liu, A. Y. (2013, Septiembre 19). *HELDA - Digital Repository of the University of Helsinki.* Retrieved from https://helda.helsinki.fi/bitstream/handle/10138/40769/technical-report-C-2013-1.pdf

Apéndices

Apéndice A. Instalación de otros controladores que son de utilidad

Estos controladores se utilizaron para pruebas internas que nos ayudaran a encontrar información de ellos y dar una mejor selección sobre cual podíamos utilizar en el proyecto. Se muestra la forma en el cual se instalaron para su utilización

1.1 Instalación de HP Virtual Application Networks

Para dicha instalación se requiere verificar la versión disponible en el sitio web, en nuestro caso se trabajara con la versión 2.5.15 disponible en https://h10145.www1.hp.com/downloads/DownloadSoftware.aspx?SoftwareReleaseUId=12097&ProductNumber=J9863AAE&lang=&cc=&prodSeriesId=&OrderNumber=&PurchaseDate=

Luego de ello procedemos a descargar el archivo, esto demora unos minutos, dentro del .rar descargado se encontrarán cuatro archivos, los cuales son:

1. hp-sdn-apidoc-2.5.15
2. hp-sdn-ctl_2.5.15.1175_amd64
3. hp-sdn-sdk-2.5.15
4. HP-VAN-SDN-Controller-2.5.15-Release-Notes

Además, se encontrará en la página web el manual de instalación para dicho software, por tanto, se procede a la instalación.

~$ sudo apt-get update

~$ sudo apt-get install python-software-properties

~$ sudo apt-get install ubuntu-cloud-keyring

~$ sudo add-apt-repository cloud-archive:icehouse

~$ sudo apt-get update

~$ sudo apt-get install keystone

~$ sudo dpkg --unpack hp-sdn-ctl_2.8.8.0366_amd64.deb #en nuestro caso

NOTA: cabe resaltar que si el último comando no funciona es debido a los requerimientos mínimos del software por lo tanto si no se cumplen este provocará un error. Si no se cuenta con los requerimientos mínimos se puede evadir esta verificación del software de la siguiente manera **touch /tmp/override.txt**. Así pues, se ejecutará el último comando y se instalará el software.

~$ sudo apt-get install -f

Verificando la instalación

Se ejecuta el siguiente comando **sudo dpkg -l hp-sdn-ctl** y aparecerá lo siguiente:

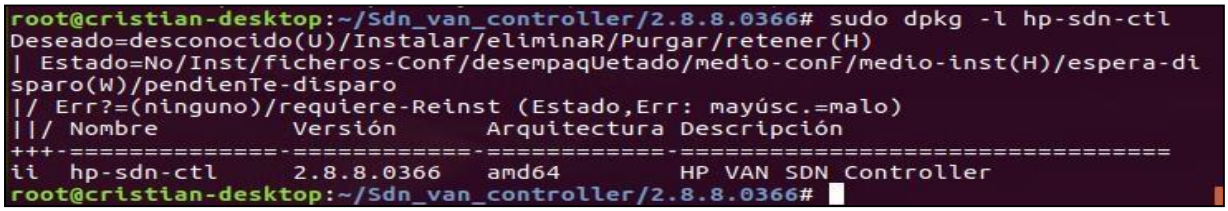

Figura 43. Comprobación de instalación sdn_van_controller,

Fuente: Autor del proyecto

Si en llegado caso el indicado es diferente de ‘ii’ quiere decir que el controlador está instalado de forma incorrecta.

Este controlador utiliza unas versiones recientes compatible con los sistemas operativos recientes, pero no es de carácter libre por lo tanto no se incluye en este análisis.

1.2 Instalación TREMA controller

Para la instalación se esté controlador es necesario clonar el repositorio de git que se encuentra https://github.com/trema/trema luego procedemos con la instalación.

~$ gem install trema #si no se instala por permisos acceder como root

~$ cd trema

~$ bundle install

De esta manera el controlador se encuentra instalado en su máquina, para comprobar la instalación se ejecuta el comando **~$ bundle info trema** y aparecerá lo siguiente.

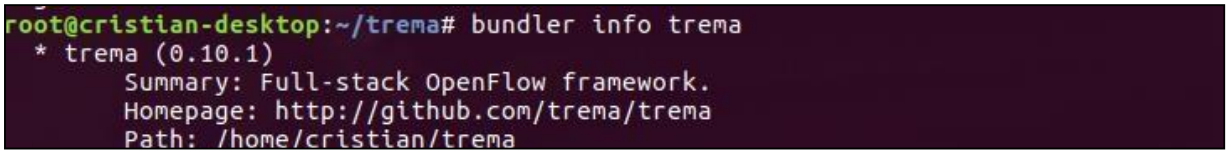

Figura 44. Comprobación de instalación Trema controller,

Fuente: Autor del proyecto

1.3 Instalación Opendaylight

Debido a que este controlador se basa en Beacon y este a su vez está desarrollado con java cabe resaltar que se necesita la última versión de java en este caso java 8, luego de tener instalado java en su ordenador se descarga el archivo de Opendaylight en este caso se selecciona la versión Karaf **Carbon,** en un ".tar", se descomprime. con esto es suficiente para obtener el controlador de Opendaylight.

Para la comprobación se ejecuta el siguiente comando. **/bin/karaf** lo cual debe aparecer algo como esto:

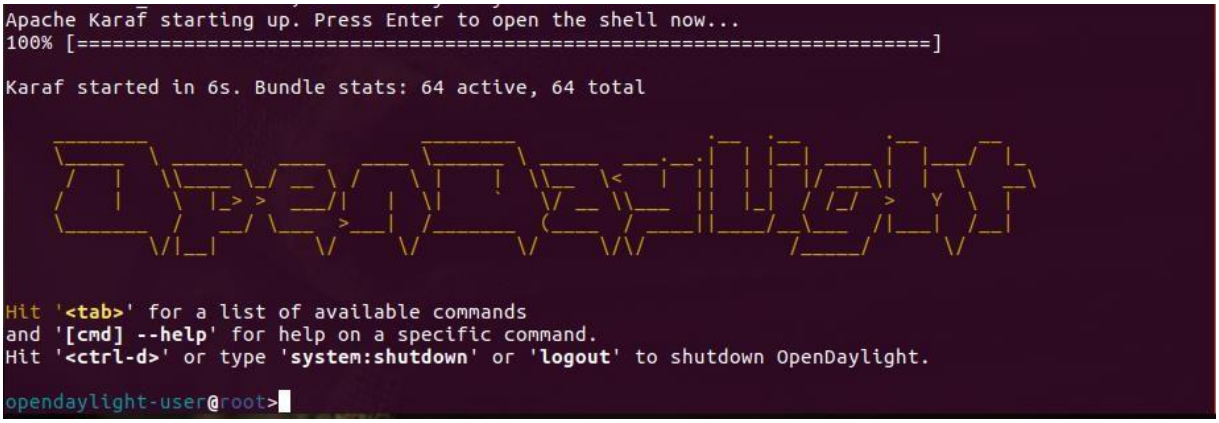

Figura 45. Comprobación de instalación OpenDaylight.

Fuente: Autor del proyecto

Si es necesario instalar características al controlador se realizará de la siguiente manera

1. Revisar características instaladas: **feature:list -i**
2. Revisar componentes disponibles: **feature:list**
3. Instalar characteristic deseada: **feature:install <feature1><feature2> ... <featureN-name>.**

Para adicionar la interfaz gráfica se instalan los siguientes items:

1. odl-dlux-core
2. odl-dluxapps-nodes
3. odl-dluxapps-topology
4. odl-dluxapps-yangui
5. odl-dluxapps-yangvisualizer
6. odl-dluxapps-yangman

Apéndice B. Herramientas con las que cuenta Openflow en la versión 1.3 con RYU controller

Protocolos anexos a Ipv6 1.3

Tabla 14.

Cabeceras IPv6. Fuente Autor del proyecto

Protocolo	Dependencia	Nombre	OXM	Mask type
ipv6	ETH_TYPE = 0x08dd	DSCP	IP_DSCP	
		ECN	IP_ECN	
		Protocol	IP_PROTO	
		Source Address	IPV6_SRC	bitmask
		Destination Address	IPV6_DST	bitmask
		Flow Label	IPV6_FLABEL	bitmask
		Extension Header	IPV6_EXTHDR	bitmask
ICMPv6	IP_PROTO = 58	Type	ICMPV6_TYPE	
		Code	ICMPV6_CODE	
		ND Target Address	IPV6_ND_TARGET	
		ND Link-Layer Source	IPV6_ND_SLL	
		ND Link-Layer Target	IPV6_ND_TLL	

Fuente: Autor del proyecto

Tabla 15

Atributos de cabecera . Fuente Flowgrammable 2018

Atributo	Descripción
nxt	Encabezado siguiente
size	La longitud del encabezado de autenticación en palabras de 64 bits, restando 1.
spi	Índice de parámetros de seguridad
seq	Número de secuencia
data	Datos de autenticación

Fuente: Autor del proyecto