ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA EN ELECTRÓNICA TELECOMUNICACIONES Y REDES

"DESARROLLO DE UNA PLATAFORMA PARA EVALUAR CALIDAD DE SERVICIOS (QoS) EN REDES DEFINIDAS POR SOFTWARE (SDN)"

TRABAJO DE TITULACIÓN: **PROYECTO TÉCNICO**

Para optar el Grado Académico de:

INGENIERA EN ELECTRÓNICA, TELECOMUNICACIONES Y REDES

AUTORA: GUERRERO MAZÓN GEOVANNA DANIELA

TUTOR: ING. ALBERTO ARELLANO AUCANCELA.

Riobamba-Ecuador

2017

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA EN INGENIERÍA EN ELECTRÓNICA TELECOMUNICACIONES Y REDES

El Tribunal del trabajo de titulación certifica que: El trabajo de titulación: **DESARROLLO DE UNA PLATAFORMA PARA EVALUAR CALIDAD DE SERVICIOS EN REDES DEFINIDAS POR SOFTWARE**, de responsabilidad de la señorita Geovanna Daniela Guerrero Mazón, ha sido minuciosamente revisado por los Miembros del Tribunal del trabajo de titulación quedando autorizada su presentación.

ING. WASHINGTON LUNA **DECANO DE LA FACULTAD DE INFORMÁTICA Y ELECTRÓNICA**	_____	_____
ING. FRANKILN MORENO **DIRECTOR DE LA ESCUELA DE INGENIERIA ELECTRÓNICA, TELECOMUNICACIONES Y REDES**	_____	_____
ING. ALBERTO ARELLANO **DIRECTOR DE TESIS**	_____	_____
ING. OSWALDO MARTÍNEZ **MIEMBRO DEL TRIBUNAL**	_____	_____

Yo, Geovanna Daniela Guerrero Mazón, declaro ser la autora del presente trabajo de titulación: "DESARROLLO DE UNA PLATAFORMA PARA EVALUAR CALIDAD DE SERVICIOS (QoS) EN REDES DEFINIDAS POR SOFTWARE (SDN)", fue elaborado en su totalidad por mí, bajo la dirección del Ingeniero Alberto Arellano Aucancela, haciéndome totalmente responsable por las ideas, criterios, doctrinas y resultados expuestos en este Trabajo de Titulación, y el patrimonio de la misma pertenece a la Escuela Superior Politécnica de Chimborazo.

Geovanna Daniela Guerrero Mazón

DEDICATORIA

A Dios por haberme permitido llegar hasta este punto y haberme dado la salud para lograr mis objetivos, además de su infinita bondad y amor. A mi papito José y mamita Carmen, por mi pilar fundamental y haberme apoyado en todo momento, por sus consejos, sus valores, por los ejemplos de perseverancia y constancia que los caracterizan, por la motivación constante que me ha permitido ser una persona de bien y por el valor mostrado para salir adelante, pero más que nada, por su amor. A mi esposo Rodrigo y mi hija Danna Rafaela, les dedico mi esfuerzo y dedicación para poder culminar mi objetivo ya que son mi fuente de inspiración y amor. A mis hermanas Vale, Sofi y Reina de quienes han estado acompañándome en los buenos y malos momentos y han participado directa o indirectamente de esta tesis.

AGRADECIMIENTO

El más sincero agradecimiento a mis padres ya que fueron mi base fundamental para conseguir mi objetivo, además de brindarme su amor incondicional, también agradezco a Rodrigo por acompañarme en todo momento desde nuestra unión, a mis hermanas y al ingeniero Alberto Arellano quien me ha guiado de manera muy amable en mi trabajo de titulación.

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

ÍNDICE DE ABREVIATURAS

API	Aplication Programming Interface (Interfaz de Programación de Aplicaciones)
ARP	Address Resolution Protocol (Protocolo de resolución de direcciones)
BSD	Distribución de Software Berkeley
GUI	Interfaz Gráfica de Usuario
HP	Hewlett-Packard
HTTP	Hypertext Transfer Protocol (Protocolo de transferencia de hipertexto)
ICMP	Internet Control Message Protocol (Protocolo de Mensajes de Control de Internet)
NAC	Control Access Network (Control de acceso a la red)
NAT	Translation Address Network Traducción de direcciones de red
OF	OpenFlow
ONF	Fundation Open Networking (Fundación Open Networking)
PCP	Priority Code Point (Punto de código prioritario)
QoS	Calidad de Servicio
TCP	Transmission Control Protocol (Protocolo de control de transmisión)
SDN	Network Defined Software (Red Definida por Software)
SO	Sistema Operativo
SSH	Secure Shell (Cubierta segura)
TIC	Radio Frecuencia
TCAM	Método de acceso a las telecomunicaciones
UHF	High Frequency (Frecuencia Ultra Alta)
VLAN	Redes de área local virtuales

RESUMEN

Se desarrolló una plataforma en redes definidas por software utilizando Mininet, herramienta que permitió configurar un entorno de emulación de redes dentro de un mismo sistema, acompañado del controlador Ryu, ya que contiene scripts escritos en lenguaje Python que ayudaron a la evaluación de calidad de servicio. La metodología que se utilizó en este trabajo fue definir los conceptos más relevantes de SDN. Las herramientas que se utilizaron para este proyecto son: SO Ubuntu-14.04.3, Servidor Putty, Servidor Xming 6.9.0.38, para la implementación de la lógica se usó Ryu, para la visualización de la red el emulador Mininet 2.2.2, y Wireshark, todo en conjunto permitieron crear una plataforma de red capaz de modificar su comportamiento de forma dinámica, garantizando calidad de servicio de extremo a extremo. Se documentó la simulación de una aplicación Calidad de Servicio (QoS) sobre el controlador Ryu definiendo escenarios que abordaron funcionalidades distintas, como cambiar el nivel de prioridad de tráfico de paquetes entre hosts o también aceptar o bloquear tráfico Protocolo de mensajes de control de Internet (ICMP). A través de las pruebas realizadas mediante líneas de comando ejecutadas en el controlador, se determinaron resultados como: pruebas de conectividad entre dispositivos mediante ping, capturas de tráfico OpenFlow en Wireshark, y se verificó el correcto funcionamiento del cambio de propiedades de los paquetes en el 100% de las pruebas realizadas en QoS sobre Ryu. El cambio de propiedades que se aplicó a la red implementada mediante el controlador, demuestra la eficiencia dinámica con la que se programa la lógica de la red. Se recomienda que para futuras investigaciones se implemente en dispositivos físicos emulando un entorno real aumentando la automatización.

Palabras Claves: <TECNOLOGIA Y CIENCIAS DE LA INGENIERIA> <REDES DE COMPUTADORES> < REDES DEFINIDAS POR SOFTWARE (SDN)>, < CONTROLADOR RYU>, < CALIDAD DE SERVICIO>, <PROTOCOLO OPENFLOW>, <SIMULADOR DE REDES DE COMPUTADORES>, <NIVELES DE PRIORIDAD>.

SUMMARY

A platform in software-defined networks was developed by using Mininet, a tool which allowed to set a network emulation environment within a common system, accompanied of the Ryu controller, since it contains scripts in Python language which helped in the service quality assessment. The methodology used in this work started with the definition of the most relevant concepts of SDN. The tools used for this project are: SO Ubuntu-14.04.3, Putty server, Xming 6.9.0.38 server, for logics implementation Ryu was used, for the net visualization Mininet 2.2.2 emulator and Wireshark, all in group allowed to create a network platform able to modify its behavior dynamically, guaranteeing so the service quality from side to side. The simulation of a Quality of Service (QoS) application was documented about Ryu controller defining scenarios which coped with different functionalities, as changing the level of package traffic priorities among hosts or also accepting or blocking traffic Internet Control Message Protocol (ICMP). By means of the developed tests through commanding lines developed in the controller, there were determined results as: connectivity tests among devices through ping, traffic shots OpenFlow in Wireshark, and the correct functioning of the change of properties of the packages were verified in 100% of the tests performed with QoS about Ryu. The change of properties through the controller to an implemented network demonstrates the dynamic efficiency the net logics is programmed. It is recommended for future research to implement physical devices emulating a real environment increasing automation.

Keywords: <ENGINEERING TECHNOLOGY AND SCIENCE>, < COMPUTER NETWORKS>, < SOFTWARE-DEFINED NETWORKS>, < RYU CONTROLLER>, <QUALITY OF SERVICE>, < OPENFLOW PROTOCOL>, < NETWORK SIMULATOR>, < LEVELS OF PRIORITY>

INTRODUCCIÓN

Durante los últimos años las Redes Definidas por Software (SDN), ha sido uno de los temas más interesantes en el área de redes, ya que las grandes organizaciones se encuentran bajo presión de ser más agiles y eficientes en el enfoque tradicional de networking y como se señala en lo habitual la red tradicional de datos ha estado en gran medida centrada en hardware. Sin embargo, el interés creciente en los centros de datos definidos y virtualización han liderado una mayor dependencia de la funcionalidad de red basada en el software.

SDN (Software Defined Networking), proporciona una arquitectura de red flexible y escalable que permite adaptarse a las necesidades, y poder actuar de forma dinámica a los recursos demandados por las aplicaciones, dada esta situación ha surgido un enfoque que pretende mostrar una nueva arquitectura para las redes actuales que resuelve inconvenientes que poseen las redes tradicionales. Como resultado, en entornos de red tan complejos como los existentes en centros de datos, proporciona una simplificación de la gestión de la red, y por el consiguiente una mejor utilización de la infraestructura de la que se dispone. (Pinilla, 2015, p.6)

El presente trabajo de investigación pretende demostrar mediante virtualización el comportamiento de las redes SDN, familiarizarnos con los entornos de trabajo, apoyar el movimiento dinámico, replicación y asignación de recursos virtuales, aliviar la carga administrativa de la configuración y la provisión de funcionalidades tales como QoS y seguridad. Se describen conceptos más referentes a las Redes Definidas por Software, además también se describen características técnicas principales del Protocolo OpenFlow, del controlador Ryu, del módulo Open vSwitch.

Este proyecto será un aporte para las ramas de tecnología y redes, las metodologías que se implementarán en esta investigación son: investigación bibliográfica, desarrollo de software y hardware, documentación del proyecto. La presente investigación propone incentivar el estudio y experimentación con SDN, la información recopilada puede ser utilizada en prácticas de laboratorio.

ANTECEDENTES

Las tecnologías de la información, la computación, y la electrónica al servicio de la información irrumpen en nuestras vidas a partir de la segunda mitad del siglo XX ya que desde siempre, el hombre ha tenido la necesidad de comunicarse con los demás, de expresar pensamientos, ideas, emociones, haciendo uso generalizado de nuevas tecnologías de la comunicación e información como son computadoras, equipos multimedia, redes locales, Internet, televisión digital, telefonía móvil en actividades cotidianas referentes a transacciones económicas y también la gestión interna de empresas e instituciones.

Las redes informáticas han ido evolucionando gracias a la tecnología avanzada y en los últimos años la adopción de dispositivos de red virtualizados y el interés creciente en los centros de datos definidos, han liderado un movimiento hacia una mayor dependencia de la funcionalidad de red basada en el software impulsado en gran medida por la necesidad de una mayor agilidad.

En la actualidad las Redes Definidas por Software son de gran auge, ya que parte de una larga historia en la que el objetivo es tratar de hacer que las redes informáticas sean programables. La Fundación Open Networking, creada hace poco más de dos años, es el cuerpo líder como otras fundaciones independientes en las TIC, afirma que la actual "arquitectura de red tradicional es inadecuada para satisfacer las necesidades de empresas, operadores y usuarios finales."

Según Ignasi Errando, director técnico en Cisco España relata su definición diciendo que SDN es una evolución tecnológica que facilita una red más abierta y programable y que permite acceder mejor a ciertas funciones inteligentes. (Marca El Futuro Del Networking, 2017a)

Por otro lado Pedro Martínez Bustos, Jefe de Producto de la unidad de negocio de HP Networking relato que HP es la única compañía que proporciona a las empresas una solución de red definida por software completa, que automatiza las tareas de configuración manual a través de hardware, software y aplicaciones, desde el centro de datos hasta el escritorio, a través de un único plano de control. (Marca El Futuro Del Networking, 2017b)

FORMULACIÓN DEL PROBLEMA

Desde hace millones de años ha existido la necesidad de comunicarse y es que gracias al avance de la tecnología, hoy en día el intercambiar información se realiza de manera sencilla y fácil, ya que se desarrollaron distintos métodos para lograrlo, pasando desde sistemas rudimentarios hasta llegar a aquellos de gran complejidad y alcance. Para conseguir el crecimiento de las comunicaciones, se ha trabajado desde hace muchos años en el área de redes de manera eficaz hasta la actualidad, pero debido a que fueron construidas hace varios años ya no son capaces de satisfacer los requerimientos que se presentan hoy en día, como: la necesidad de implementar servicios de red personalizados, flexibilidad, escalabilidad, acceso rápido y cambios en los patrones de tráfico de forma dinámica, ágil a los servicios de la nube, etc.

En este ámbito las redes se han convertido en la gran posibilidad de implementar de forma alterna herramientas de software que se adapten como funcionalidad en los sistemas de redes de la actualidad, de esta forma se tornará más agradable y óptima para la administración de recursos que se encargan de controlar el flujo de datos. En donde se requiere contar con elementos cuya infraestructura sea lo suficientemente idónea para la administración de los datos de una forma eficiente, de la misma manera la necesidad de poder optimizar la transmisión de información de una forma ágil y masiva.

Es por ello que las redes han tomado un nuevo paradigma, las Redes Definidas por Software que facilita y mejora la transmisión de información en tiempo real, además la alternativa de usar aplicativos API en lugar de depender de los parámetros estándares de los fabricantes de dispositivos de comunicación. En las redes actuales, tanto el plano de control como el plano de datos están situados en los dispositivos de la red, mientras que en las redes definidas por software (SDN) el plano de datos está separado del plano de control, por lo que la toma de decisiones ya no se realiza por el dispositivo sino por el administrador de la red.

Referente a la programación de la red para implementar calidad de servicio existe una gran diferencia entre las redes actuales y las redes definidas por software (SDN), puesto que las redes actuales no se pueden programar mediante aplicaciones, y cada elemento debe ser configurado por separado, mientras que en las redes definidas por software (SDN) la red se puede programar por medio de reglas de calidad de servicio que permitan cambiar la prioridad de los flujos, permitir o denegar cualquier tipo de tráfico y haciendo uso del controlador que puede exponer interfaces de aplicación para la manipulación de la red.

JUSTIFICACIÓN TEÓRICA

El crecimiento del ancho de banda, hoy en día es un problema tal como si antes una red tenía dos o tres nodos, ahora existe la posibilidad de que llegue a tener 2000 o más, además esta red requerirá de 2000 sistemas de configuración, de esta manera el grado de complejidad y el posible error por tratamiento humano incrementa de una manera exponencial, obligando así a los prestadores de servicios a buscar estrategias para que la programación no se haga nodo a nodo, si no que por el contrario realizarlo de manera centralizada.

Se pretende que las redes definidas en software tengan visión total de la red y alta disponibilidad, el término SDN hace referencia a un nuevo modelo de arquitectura de red, esto permite a los administradores definir completamente todos los parámetros y comportamientos de la red sin depender del proveedor. Es aquí donde el administrador de la red puede definir completamente las políticas de calidad establecidas y gestionarlas en todo momento en función de las necesidades entregando optimización a fallos, actualizaciones sin alterar flujos de datos, y también la administración del propietario de manera absoluta sin limitaciones.

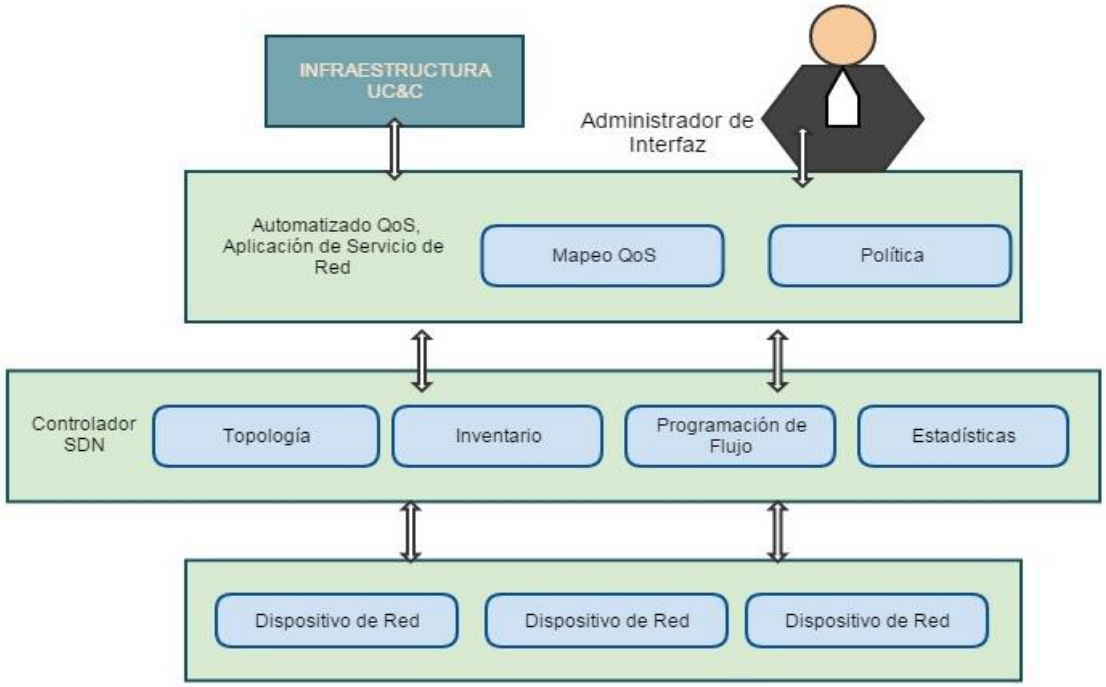


Figura 1-1 Automatización de red SDN
Fuente: GUERRERO., Daniela, 2017

JUSTIFICACIÓN APLICATIVA

En la Escuela Superior Politécnica de Chimborazo en el año 2015, Catherine Elena Yánez Carrera y Fabián Gustavo Gallegos Pillajo, realizaron el proyecto denominado "Implementación de un prototipo de Red Definida por Software para el HOTSPOT-ESPOCH mediante un controlador basado en OpenFlow". (Yanez, 2015, p.3) En el cual tiene como objetivo estudiar características de los controladores POX, RYU y PYRETIC para el desarrollo de redes virtuales definidas por software, con ello realizan la comparación de controladores en escenarios de prueba, a la vez ejecutan los controladores en la red virtual, comparando los controladores a través de indicadores para obtener el controlador más óptimo. Con éste, se explica su programación, características y funciones y se realizan pruebas para la demostración de la hipótesis. Llegando a la conclusión que el controlador POX es el más óptimo para el desarrollo de la red SDN HOTSPOT-ESPOCH con 87%, porque posee un mejor rendimiento de la red a diferencia de RYU con 80% y PYRETIC con 73% que tienen un rendimiento menor.

En el año 2013 en la Escuela Politécnica Nacional, Juan Carlos Chico Jiménez realizó el proyecto de titulación denominado "Implementación de un prototipo de una red definida por software (SDN) empleando una solución basada en hardware" (Chico, 2013, p.125) El cual consiste en el diseño e implementación de una red SDN. Para la realización del proyecto se utilizan diferentes herramientas como es el caso del software Mininet, el mismo que incluye el protocolo de comunicaciones OpenFlow necesario para la simulación de este tipo de redes, y posteriormente se trabaja con un controlador diferente al que viene instalado por defecto en el software de simulación. Mediante el cual se añade más funcionalidades y gestiona de manera personalizada el tráfico y los requerimientos de la red. En la etapa de diseño y simulación se evalúan los resultados obtenidos con el objetivo de realizar la implementación del prototipo físico. Determinando así la topología que se utilizará, los componentes que compondrán la red y el software de programación necesario para realizar los distintos cambios en el controlador en dependencia de los requerimientos de la red, con lo cual se corroboró las 6 ventajas que presentan las redes definidas por software ya que permiten modelar la red una manera ágil y personalizada además que proveen una visión global de la misma.

Otro proyecto similar se realizó en el año 2014, en la Escuela Politécnica Nacional por Diana Gabriela Morillo Fueltala denominado "Implementación de un prototipo de una red definida por software (SDN) empleando una solución basada en Software". (Morillo, 2014, p.13) En el cual

se implementa una red SDN utilizando una infraestructura basada en software, debido a esto se reemplazan los equipos físicos por máquinas virtuales. Para el desarrollo del prototipo de red se utilizaron máquinas virtuales para emular a los host de la topología y Open vSwitch para emular un switch OpenFlow, además se desarrolló una aplicación NAC (Network Access Control) mediante la cual se permite o deniega el acceso a la red, siendo Floodlight y Trema los controladores utilizados para el desarrollo para el desarrollo de las aplicaciones SDN. Al finalizar el proyecto se concluye que: las redes definidas por software son ideales para el control de tráfico en las redes ya que hacen posible la implantación de un sin número de aplicaciones. Se corroboró además que la implementación de equipos virtuales para el desarrollo de la infraestructura SDN es una solución mucho más económica y brinda las mismas características y potencia que una infraestructura física.

OBJETIVOS

OBJETIVOS GENERALES

Desarrollar una plataforma para evaluar Calidad de Servicios en Redes Definidas por Software

OBJETIVOS ESPECÍFICOS

- Determinar los requerimientos para evaluar calidad y servicio de la red definida por software.
- Analizar los requisitos técnicos, limitaciones y ventajas del emulador Mininet.
- Evaluar mecanismos en un entorno de red emulado para generar un ambiente de trabajo de Redes Definidas por Software.
- Realizar una banda de pruebas mediante virtualización para verificar parámetros de QoS y el comportamiento de redes definidas por software en la plataforma implementada.

CAPÍTULO I

1 MARCO TEÓRICO

1.1 Redes Definidas por Software

1.1.1 *Definición*

Las Redes Definidas por Software (SDN) es una arquitectura emergente que es dinámica, manejable, rentable y adaptable, lo que lo hace ideal para el alto ancho de banda, la naturaleza dinámica de las aplicaciones actuales. Básicamente tienen como finalidad entregar el control y la inteligencia de los dispositivos de la red a un solo servidor externo o controlador, esta iniciativa fue creada para funcionar en la era de la nube, ya que producen cambios importantes en cómo se construyen las redes ya que desprende el concepto de control con hardware, y éste se otorga a una aplicación de software o controlador. (Software-Defined Networking, 2017, https://www.opennetworking.org/).

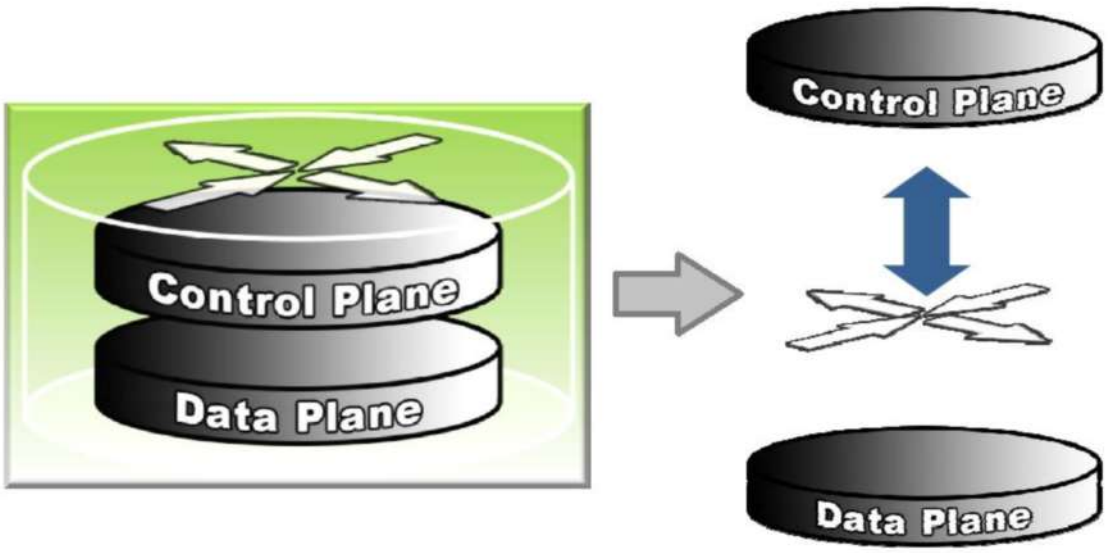


Figura 2-1: Separación Plano de Control y Plano de Datos
Fuente: www.opennetworking.org

El servidor se encargará de manejar el tráfico a través de entradas que son las aplicaciones de los usuarios que le dirán a los equipos qué camino tomar seleccionando la mejor ruta constantemente, a diferencia de la arquitectura tradicional de red, esto permite tener una visión y control global de

la red, permitiendo definir el flujo de información, la personalización y modificación de dicha infraestructura de acuerdo a las necesidades del usuario final.

En las grandes empresas de hoy en día, donde se han incrementado el crecimiento de las conexiones, y la estructuración de los datos, las SDN se ha convertido con el tiempo en una estrategia para enfrentar las necesidades que se observan a nivel de redes, una de las características más importantes es el uso de arquitecturas de código abierto.

El protocolo OpenFlow se ha creado para asegurar la interoperabilidad que conforma una red de datos como funcionalidad a los switches, routers y puntos de acceso inalámbrico, además de contar con los recursos físicos para implementar esta nueva tendencia de redes.

1.1.2 *Arquitectura SDN*

En la **Figura 3-1** se observa la arquitectura SDN que está compuesta por tres capas: Capa de Aplicación, Capa de Control, Capa de Datos.

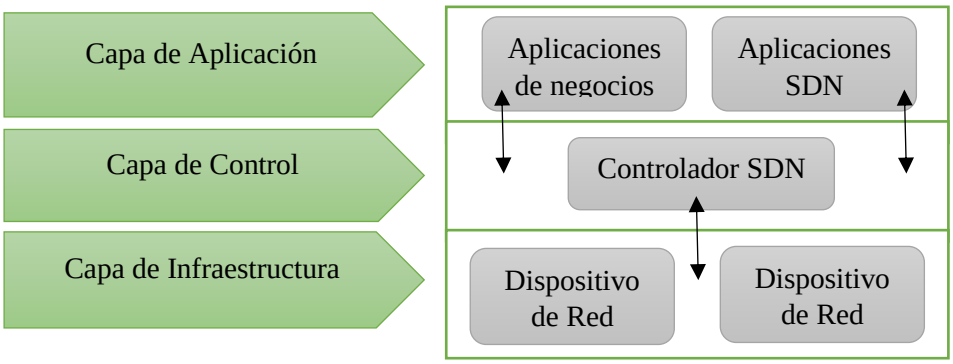


Figura 3-1: Arquitectura SDN
Fuente: www.opennetworking.org

1.1.2.1 *Capa de Aplicación*

Consiste en las aplicaciones de negocio de los usuarios finales. Esta capa automatiza tareas de configuración, provisión y despliegue de nuevos servicios en la red, se comunica con la capa de Control por medio de un API (Application Programming Interface) para tener una visión global de las condiciones actuales de la red, con el fin de mejorar la transmisión de datos y mejorar la toma de decisiones locales por nodos de redes individuales acerca de cómo tratar los flujos de tráfico de una aplicación en particular y cómo se deben compartir con las plataformas de control. Ofrece a los operadores nuevas vías de ingresos, diferenciación e innovación de esta forma mejorar la experiencia de usuarios que exigen alta calidad con aplicaciones multimedia en tiempo

real y los servicios en la nube. Permitiendo a los servicios y aplicaciones simplificar y automatizar las tareas de configuración, provisión y gestionar nuevos servicios en la red.

1.1.2.2 *Capa de Control*

El plano de control y de datos en una red tradicional se encuentra en un mismo dispositivo de red, por lo que la flexibilidad y escalabilidad para la introducción de nuevas funcionalidades se vuelve limitada. SDN, es una tecnología que tiene el plano de control separado del plano de datos, como se indicó en la **Figura 3-1**, permitiendo la adaptación de nuevos servicios de acuerdo a las necesidades de los usuarios. El plano de control o controlador, es el que ejerce la función principal de control en una SDN, es el encargado de la configuración de cada nodo, manejar los dispositivos de red y organizar el envío de información. Al comparar con una red tradicional, el administrador de red tendría que efectuar configuraciones en cada dispositivo de forma manual. Se definen dos tipos de interfaces para la comunicación entre la capa de control, aplicación e infraestructura, estas son: Northbound y Southbound. La interfaz Northbound permite desarrollar aplicaciones de alto nivel, como, la provisión de sistemas de seguridad, integración de middlebox, recursos para la administración y control, etc. En cambio, la interfaz Southbound permite la comunicación entre el plano de datos y plano de control. Existen diferentes interfaces Southbound, tal como OpenFlow, el protocolo Juniper's Contrail Controller y Cisco's Open Network Environment. (Ibañez y Lévano y Nieto, 2015: p.14)

Existe una amplia variedad de controladores, como, por ejemplo, Opendaylight, Ryu, ONOS, Floodlight, entre otros. En la **Tabla 1-1** se muestra y se detalla los principales controladores clasificados según el lenguaje de programación en el que han sido desarrollados.

Tabla 1-1: Clasificación más importantes de Controladores SDN.

Lenguajes de programación	Controladores
C++	NOX
Java	Floodlight, OpenDayLight
Python	POX, Ryu

Realizado por: Daniela Guerrero, 2017
Fuente: Dr. Nick Feamster, Software Defined Networking

1.1.2.3 *Capa de Datos*

Es el encargado de procesar los paquetes que llegan a los dispositivos de red a través del medio físico. La información se envía hacia el controlador, para que efectúe cambios en la cabecera convenientemente y realice las funciones de procesamiento o el análisis de datos, posteriormente

la información ingresa y egresa del plano de datos a través de puertos sean estos virtuales y/o físicos. El controlador puede utilizar algoritmos programados previamente en el plano de control y de esta forma definir de forma clara y sencilla una gran cantidad de funcionalidades nuevas, que con la red tradicional sería difícil de implementar. El plano de datos debe realizar consultas al controlador para realizar un reenvío ya que no éste no puede realizarlo de manera autónoma, sin embargo en casos, como, por ejemplo, en los cuales tenga que enfrentar una falla en la red, el controlador establece ordenes preestablecidas para que el plano de datos los utilice. Un controlador SDN permite que gestione un amplio rango de recursos de plano de datos, lo cual ofrece el potencial de unificar y simplificar su configuración, como se muestra en la **Figura 3-1.**

1.1.3 *Fundación Open Networking*

La Fundación Open Networking (ONF) es el que está más relacionado con el desarrollo y estandarización de SDN y fue innovada para la adopción de esta nueva tecnología, e implementar SDN a través de estándares abiertos sin fines de lucro, con normas que son necesarias para la aceleración de la industria de las redes. Con el fin de hacer de SDN una realidad comercial que cumpla con las necesidades del cliente, la ONF desarrolló estándares abiertos tales como la Norma OpenFlow y la configuración y gestión OpenFlow Protocolo Estándar. Los grupos de trabajo ONF también están allanando el camino para el desarrollo de soluciones interoperables mediante la colaboración con los principales expertos del mundo en la SDN y OpenFlow sobre conceptos, los marcos, la arquitectura y estándares. El Standard OpenFlow es el primer y único interfaz de comunicaciones estándar independiente del proveedor definido entre las capas de control y la transmisión de una arquitectura SDN. (SDN Architecture, 2014. Disponible en: http://www.ramonmillan.com/tutoriales/sdnredesinteligentes.php)

1.1.4 *Ventajas de una red SDN*

1.1.4.1 *Escalabilidad*

Posibilita manejar el control de la red mediante la implementación de programas que optimizan rápidamente los recursos de una red SDN dando como resultado una red más flexible y configurable, por lo que es posible experimentar con nuevas aplicaciones, configuraciones, etc. Sin duda la agilidad y velocidad de aprovisionamiento de servicios y recursos se incrementa, además, que se puede reservar ancho de banda para nuevos servicios rápidamente.

1.1.4.2 *Virtualización de la red*

La virtualización permite tener redes dinámicas y adaptables a cambios, ya que dispone de un controlador SDN que contiene una visión global de la red y tiene la inteligencia para su control. Una red SDN se puede reconfigurar en poco tiempo, evitando molestos procesos de configuración remota de uno de los equipos de conmutación.

1.1.4.3 *Automatización*

La administración de la red se centraliza en un único punto, permitiendo ajustar el tráfico para satisfacer cambios que se puedan originar, focalizando la responsabilidad y eliminando múltiples puntos de decisión, que frecuentemente llevaban a agujeros de seguridad, errores de configuración, cuellos de botella, etc.

1.1.4.4 *Seguridad*

La seguridad en las configuraciones de los switches y routers pasa a ser controlada por el Controlador, de forma que no habrá agujeros de seguridad en la red implementada.

1.2 OpenFlow

Es un estándar creado por la Universidad de Stanford, es el primer estándar que define la interfaz de comunicación entre el controlador y los diferentes dispositivos de la red de la arquitectura SDN, diseñado para permitir a los investigadores ejecutar protocolos experimentales con el objetivo de ajustar el comportamiento del plano de datos. Dicho estándar es abierto y ha sido ampliamente implementado por los fabricantes, por lo que lo convierte en un referente en la interfaz southbound. La primera especificación surgió a finales de 2009, y desde entonces ha seguido evolucionando con la supervisión de la ONF, ya que desde el año 2011 tiene el control sobre la especificación. (Enabling Innovation in Campus Network, 2017. Disponible en: http://archive.openflow.org/documents/openflow-wp-latest.pdf)

1.2.1 *Switch OpenFlow*

La idea básica parte del hecho de que la mayoría de los switches y enrutadores Ethernet modernos contienen tablas de flujo, normalmente construido a partir de TCAM (Método de acceso a las telecomunicaciones), que se ejecutan para implementar firewalls, NAT, QoS, y para recopilar estadísticas. Entonces, un switch OpenFlow consiste en una tabla de flujo y un canal externo que se conecta al controlador. Existe un conjunto común de funciones que se ejecutan en muchos conmutadores y enrutadores, OpenFlow los explora y proporciona un protocolo abierto para programar las tablas de flujos en diferentes switches y enrutadores. La ruta de datos o datapath de un conmutador OpenFlow consiste en una tabla de flujos, y una acción asociada con cada entrada de flujo. El conjunto de acciones soportadas por un conmutador OpenFlow es extensible, es por ello que para el alto rendimiento y bajo costo del datapath debe tener un grado cuidadosamente prescrito de flexibilidad.

Esto significa renunciar a la posibilidad de especificar manipulación arbitraria de cada paquete y buscando una más limitada gama de acciones. (The Openflow Switch, 2017. Disponible en: http://archive.openflow.org/documents/openflow-wp-latest.pdf). Un switch OpenFlow está formado de tres partes, las cuales se pueden ver en la **Figura 2-1:**

Tabla 2-1: Partes de un Switch OpenFlow.

Tabla de Flujo	Que informa al switch cómo procesar el flujo con una acción asociada con cada entrada de flujo.
Canal Seguro	Que conecta el switch a un proceso de control remoto o bien llamado controlador, permitiendo el uso de comandos y paquetes para ser enviados entre dicho controlador y el switch.
Protocolo OpenFlow	El que proporciona una interfaz abierta y sirve para que el controlador se comunique con el switch.

Realizado por: Daniela Guerrero, 2017
Fuente: http://archive.openflow.org/documents/openflow-wp-latest.pdf

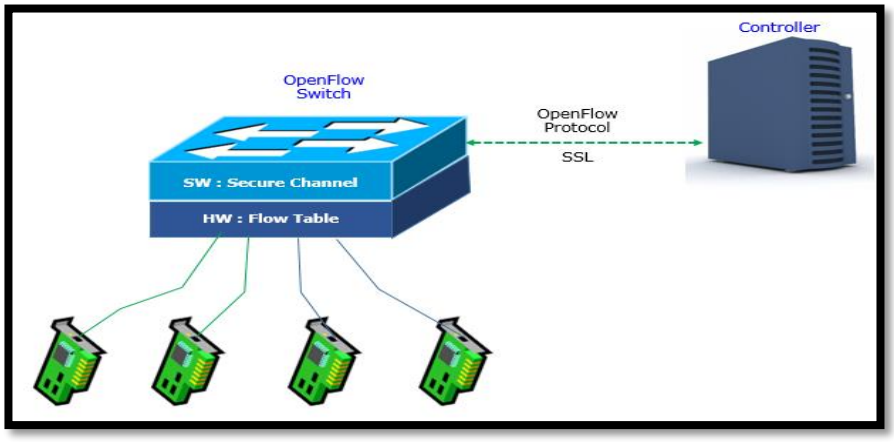


Figura 4-1: Arquitectura Switch OpenFlow.
Fuente: http://www.sharetechnote.com/html/IP_Network_SDN_OpenFlow.html

1.2.1.1 *Campos de encabezado*

Cuando se recibe un nuevo paquete entrante por alguno de los puertos del switch, se desarrolla un proceso llamado packet matching, que consiste en comparar los campos de las cabeceras del paquete y su puerto origen con los campos de correspondencia de las entradas de la tabla de flujos. (Puente, 2015a: p.47). Cuando el paquete coincide con los campos de correspondencia de una entrada de la tabla de flujos, se realiza el proceso en dicha entrada y se actualizan las estadísticas. Las acciones que se pueden realizar son:

- Reenviar el flujo a un puerto o conjunto de puertos específicos: Se trata de encaminar los paquetes a través de la red.
- Reenviar el flujo al controlador: A través del canal seguro, el controlador recibe los paquetes de un determinado flujo para procesarlos y tomar una cierta decisión al respecto.
- Modificar campos del paquete: Permite editar, eliminar o añadir campos de las cabeceras de los paquetes recibidos de un cierto flujo.
- Descartar el flujo de paquetes: Puede ser utilizado el switch OpenFlow como un firewall para implementar medidas de seguridad en la red.

La primera versión de OpenFlow muestra 12 campos para desarrollar este proceso, y se detalla en la **Tabla 3-1**:

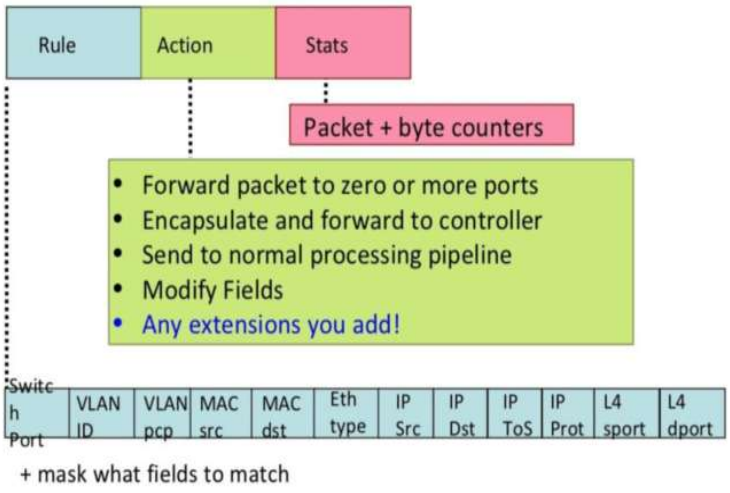


Figura 5-1: Campos de encabezado OpenFlow
Fuente: https://www.slideshare.net/openflow/openflow-tutorial

Tabla 3-1: Campos de encabezado, longitudes y forma que son aplicados en las entradas de flujo

Campo	Bits	Casos aplicables	Notas
Ingress Port	(Dependiente de la implementación)	Todos los paquetes	Representación numérica del puerto de entrada, empezando en 1
Ethernet Source Address	48	Todos los paquetes en los puertos habilitados	
Ethernet Destination Address	48	Todos los paquetes en los puertos habilitados	
Ethernet Type	16	Todos los paquetes en los puertos habilitados	
VLAN Id	12	Todos los paquetes Ethernet del tipo 0x8100	
VLAN Priority	3	Todos los paquetes Ethernet del tipo 0x8100	Campo PCP (Priority Code Point) de una VLAN.
IP Source Address	32	Todos los paquetes IP y ARP	Se le puede aplicar una máscara de subred
IP Destination Address	32	Todos los paquetes IP y ARP	Se le puede aplicar una máscara de subred
IP Protocol	8	Todo paquete IP e IP sobre Ethernet, paquetes ARP	Sólo los 8 bits menos significativos del código ARP son utilizados
IP ToS Bits	6	Todos los paquetes IP	
Transport Source Port/ICMP Type	16	Todos los paquetes TCP, UDP e ICMP	Sólo son utilizados los 8 bits menos significativos del tipo ICMP.
Transport Destination Port/ICMP Code	16	Todos los paquetes TCP, UDP e ICMP	Sólo son utilizados los 8 bits menos significativos del tipo ICMP.

Realizado por: Daniela Guerrero, 2017
Fuente: SOTELO, Marco. pg.48

1.2.1.2 *Tablas de flujos*

Un conmutador lógico OpenFlow consiste en una o más tablas de flujo y una tabla de grupo, que realizan paquetes búsqueda y reenvío, y uno o más canales OpenFlow a un controlador externo. Cada entrada de flujo se compone de los siguientes elementos de la **Tabla 4-1**:

Tabla 4-1: Componentes de una entrada en una tabla de flujos en un Switch OpenFlow.

Match Fields (Campos de coincidencia)	Priority (Prioridad)	Counters (Contadores)	Instructions (Instrucciones)	Timeouts (tiempo de espera)	Cookie

Realizado por: Daniela Guerrero, 2017
Fuente: http://opennetworking.org/images/stories/downloads/specification/openflow-spec-v1.3.0.pdf

- Campos de correspondencia: Definen un cierto flujo mediante el establecimiento de un conjunto de campos que se comparan con los paquetes recibidos en el switch.
- Prioridad: Implanta un orden de preferencia en las entradas de una tabla de flujos.
- Contadores: Muestran estadísticas sobre el flujo correspondiente a una cierta entrada, como por ejemplo el número de paquetes o bytes identificados.
- Instrucciones: Indican una acción o conjunto de acciones que se deben ejecutar con un cierto flujo.
- Temporizadores: Permiten la eliminación de una cierta entrada de forma automática transcurrido un cierto periodo de tiempo.
- Cookie: Es una información añadida por el controlador que permite la identificación de la entrada de flujo.

1.2.2 *Controlador*

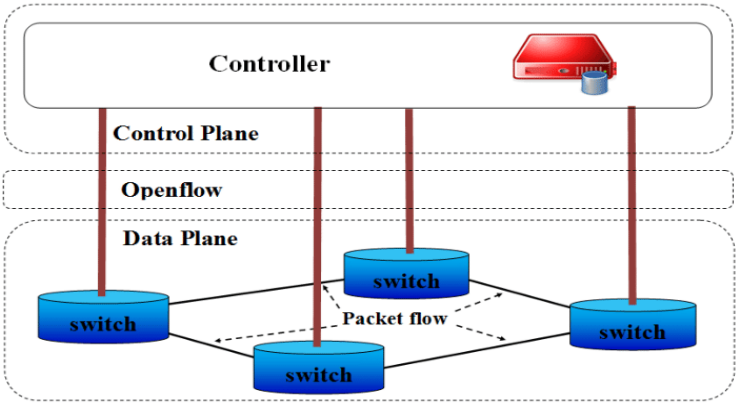


Figura 6-1: Controlador OpenFlow.
Fuente: www.cisco.com

El controlador es el que tiene toda la lógica de la red, tiene casi las mismas características que cualquier otro controlador para SDN, ya que se definen reglas para administrar el flujo de datos de la red. Es una aplicación software que hace un control centralizado ya sea física o lógicamente, que se encarga de programar en todos los dispositivos de la red las entradas de flujo para gestionar el tráfico como sea necesario. El controlador funciona como una capa que abstrae la infraestructura física, ya que esto facilita la creación de aplicaciones y servicios que gestionan la red de flujos de entradas. Esto permite una configuración más rápida que en las redes tradicionales, donde se espera a que el fabricante lo haga, al igual que la implementación de nuevas aplicaciones y servicios. (Álvarez, 2015: p. 11)

1.2.3 *Mensajes entre el switch y el controlador*

El protocolo OpenFlow desarrolla una comunicación entre los switches y el controlador sobre el nivel de transporte mediante el protocolo TCP y define los siguientes tipos de mensajes: controlador a conmutador, simétrico y asíncrono. Cada mensaje OpenFlow comienza con la misma estructura en la cabecera, lo que permite su utilización independientemente de la versión en la que se esté manejando. El protocolo OpenFlow soporta tres tipos de mensajes. (Isa, 2016: p. 6) Éstos son:

- Controlador a conmutador: Iniciado por el controlador y usado para inspeccionar y gestionar el estado del conmutador, en ciertos casos se requieren respuesta.
- Asíncronos: Es iniciado en el conmutador y dirigido al controlador sin que este le haya solicitado, aquí se actualizan la información del controlador cuando se producen eventos en la red, por ejemplo la llegada de un paquete y cambios de estado en el conmutador.
- Simétricos: Son iniciados por ambos, es decir, se transmiten ya sea por el controlador o por el conmutador sin que haya surgido ninguna solicitud previa, aquí se definen dos tipos de conmutadores: OpenFlow-only que quiere decir que según la capacidad puede trabajar solo con OpenFlow y OpenFlow-enabled que pueden procesar el paquete utilizando algoritmos tradicionales de conmutación o de encaminamiento.

1.3 Controladores SDN

El controlador es el elemento central o principal de la arquitectura SDN, ya que se trata del dispositivo que implementa toda la lógica de la red, ejecutando las reglas que le llegan de las aplicaciones para administrar el flujo de datos y distribuyéndolas entre los diferentes dispositivos de capa física de la red. Se trata de un sistema operativo que aporta como una gestión de la red, como un mecanismo de descubrimiento de dispositivos y topología, como una sesión TCP, y como un conjunto de APIs de los servicios del controlador. (Build SDN agilely, 2017. Disponible en: https://osrg.github.io/ryu/). Existen diferentes tipos de controladores que manejan distintos lenguajes de programación y plataforma, pero que en esencia realizan las mismas funciones como: comunicarse, mediante el protocolo OpenFlow, con los dispositivos de la red. Sin duda el controlador permite una configuración más rápida que en las redes tradicionales, donde se espera a que el fabricante lo haga, al igual que la implementación de nuevas aplicaciones y servicios. Entre otros controladores algunos de los más importantes son: Ryu, NOX, POX, Beacon,

Floodlight, OpenDayLight. En resumen se muestra en la **Tabla 5-1** las características de cada controlador mencionado anteriormente:

Tabla 5-1: Principales características de controladores SDN

Característica	NOX	POX	Beacon	Floodlight	OpenDayLight	Ryu
Soporte OpenFlow	OF v1.0	OF v1.0	OF v1.0	OF v1.0	OF v1.0/v1.3	OF v1.3
Lenguaje	C+ +	Python	Java	Java	Java	Python
Plataformas	Linux	Linux, Mac Os, Windows	Linux, Mac Os, Windows	Linux, Mac Os, Windows	Linux, Mac Os, Windows	Linux
Código Abierto	Si	Si	Si	Si	Si	Si
Interfaz Gráfica	Python + QT4	Python+ QT4	Web	Web	Web	Web
API	No	No	No	Si	Si	Si
Documentación	Media	Pobre	Buena	Buena	Media	Media

Fuente: GREGORIO, Eduardo. 2016. p. 17.

Para este proyecto técnico se va a utilizar el controlador Ryu, ya que proporciona aplicaciones escritas en lenguaje de programación Python, una de ellas es Calidad de Servicio la cual servirá para cumplir los objetivos de este trabajo de titulación.

1.4 Controlador RYU

El controlador Ryu es una infraestructura que 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 Controller es un controlador de red abierto (SDN), definido por software, diseñado para aumentar la agilidad de la red al facilitar la administración y la adaptación del tráfico. El controlador Ryu proporciona componentes de software, con interfaces de programa de aplicación (API) bien definidas, que facilitan a los desarrolladores la creación de nuevas aplicaciones de administración y control de redes. Este controlador está desarrollado en python, un lenguaje que posee ventajas en su sintaxis y semántica ya que es relativamente sencilla a comparación de lenguajes similares y es totalmente compatible con la programación orientada a objetos. (Tinajero, 2016a, pp. 22-23).

1.4.1 *Arquitectura Ryu*

Ryu como cualquier otro controlador puede crear y enviar mensajes OpenFlow, escuchar mensajes asíncronos como flow-removed y Packet-in, analizar los paquetes entrantes. (Tinajero, 2016b, pp. 24). En la **Figura 7-1** se muestra los elementos que constituyen la arquitectura Framework del controlador Ryu.

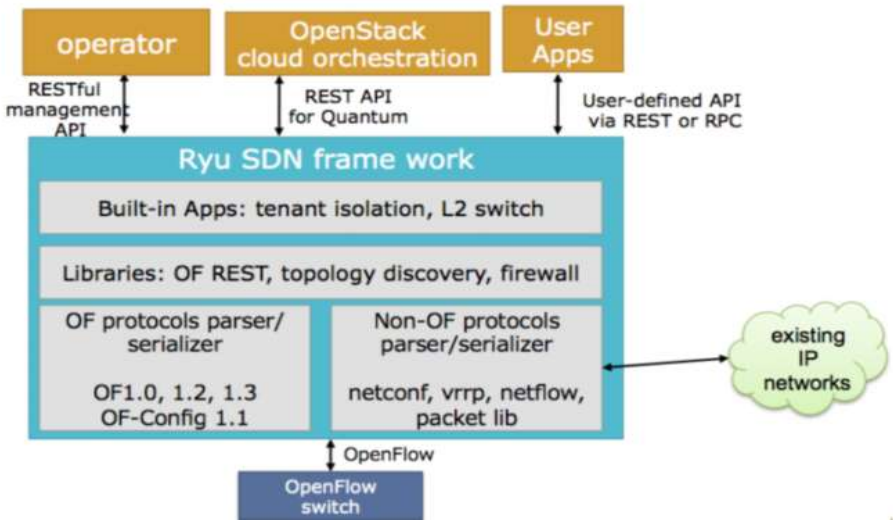


Figura 7-1: Arquitectura Controlador Ryu
Fuente: RYU OpenFlow Controller, pg. 5. Architecture

1.4.2 *Componentes Ryu*

Ryu consta de componentes útiles para aplicaciones SDN, se pueden modificar los componentes existentes e implementar unos nuevos. Los componentes incluidos en Ryu se observan en la **Tabla 6-1.**

Tabla 6-1: Componentes incluidos en el controlador Ryu

COMPONENTES RYU		
Ejecutables	bin / ryu-manager	El ejecutable principal.
Componentes de Base	ryu.base.app_manager	-Cargar aplicaciones Ryu. -Proporcionar contextos a las aplicaciones Ryu. -Ruta de mensajes entre aplicaciones de Ryu.

Controlador OpenFlow	ryu.controller.controller	Maneja las conexiones de los interruptores. Genera y enruta eventos a entidades apropiadas como aplicaciones Ryu.
	ryu.controller.dpset	Planeado para ser reemplazado por ryu/topología.
	ryu.controller.ofp_event	Definiciones de eventos de OpenFlow.
	ryu.controller.ofp_handler	Manejo básico de OpenFlow, incluida la negociación.
Aplicaciones Ryu	ryu.app.simple_switch	Una implementación de switch de aprendizaje OpenFlow 1.0 L2.
	ryu.topology	Cambiar y vincular el módulo de descubrimiento. Planeado para reemplazar ryu/controller/dpset.
Bibliotecas	ryu.lib.packet	Implementaciones de descodificadores/codificadores de protocolos populares como TCP/IP.
	ryu.lib.ovs	Ovsdb biblioteca de interacción.
	ryu.lib.of_config	OF-Config implementación
	ryu.lib.netconf	Definiciones de NETCONF utilizadas por ryu/lib/of_config.

Realizado por: Daniela Guerrero, 2017
Fuente: http://ryu.readthedocs.io/en/latest/components.html

1.4.3 *Aplicaciones Ryu*

Las aplicaciones Ryu son entidades de subproceso único que implementan varias funcionalidades en Ryu y envían mensajes de forma asíncrona entre sí, a estos se les llama eventos. Se detalla las funcionalidades de cada proceso que en la **Figura 6-1** se muestra el modelo de programación de aplicaciones.

1. Evento (Event).- Son objetos de clase que se heredan de ryu.controller.event.EventBase y es la comunicación entre aplicaciones que se realiza transmitiendo y recibiendo eventos.
2. Cola de eventos (Event Queue).- Cada aplicación tiene una sola cola para recibir eventos.
3. Hilos (Threads).- Ryu se ejecuta en multi-hilo utilizando eventlets. Debido a que los subprocesos no son preventivos, se debe tener cuidado cuando se realice procesos que consumen mucho tiempo.

4. Manejadores de eventos (Event handlers).- Puede definir un manejador de eventos decorando el método de clase de aplicación con un ryu.controller.handler.set_ev_cls decorador. Cuando ocurre un evento del tipo especificado, se llama al controlador de eventos desde el bucle de eventos de la aplicación.

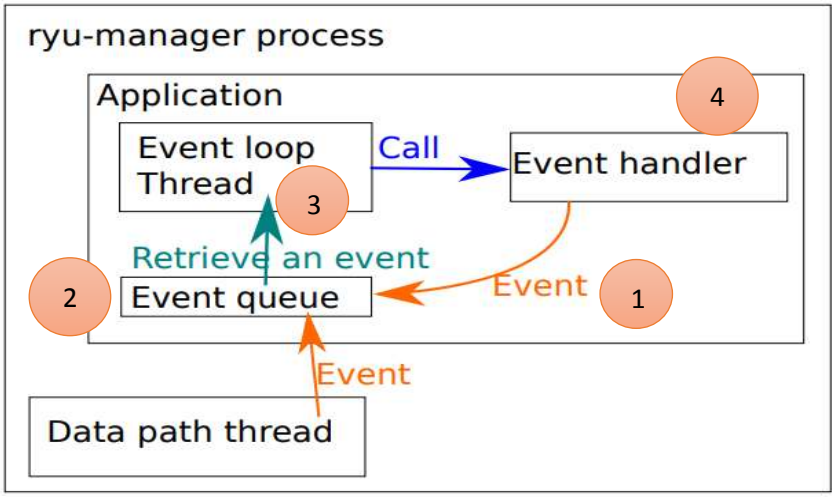


Figura 8-1: Modelo de programación de aplicaciones en Ryu
Fuente: Ryubook, pg. 181. Architecture.

Hay algunas fuentes de eventos internos de Ryu que no son aplicaciones, uno de los ejemplos de tales fuentes de eventos es el controlador OpenFlow. Además también un evento puede contener objetos arbitrarios de python, no se recomienda pasar objetos complejos, por ejemplo, objetos que no se pueden pegar entre las aplicaciones de Ryu. (Ryubook, 2017. Disponible en: https://osrg.github.io/ryu/resources.html#books)

Cada aplicación Ryu tiene una cola de recepción para eventos, esta cola se define como FIFO (primero en entrar, primero en salir) y conserva el orden de los eventos. Además también cada aplicación Ryu tiene un hilo para el procesamiento de eventos, el subproceso sigue drenando la cola de recepción al eliminar un evento y llamar al controlador de eventos apropiado para el tipo de evento, cuando se llama al controlador de eventos en el contexto del subproceso de procesamiento de eventos, debe tener cuidado al bloquear. Mientras se bloquea un controlador de eventos, no se procesarán más eventos para la aplicación Ryu. [19]

1.5 Mininet

Mininet es un software de emulación de redes que permite crear una red virtual de switches, hosts y controladores SDN, ejecutándose muy rápidamente sobre el kernel de LINUX con un simple comando. Un host de Mininet se comporta como una máquina real y es un proyecto de código

abierto. (Llaulet, 2016. pp. 32-33) Este emulador ofrece un entorno de línea de comandos simple que permite una fácil interacción del usuario con la red virtualizada, además, cuenta con una API de Python que permite la construcción y manejo de redes de datos a partir de un conjunto de líneas de código. (Henao, 2015a: p. 9)

Figura 9-1. Representación comando principal del emulador Mininet
Fuente: http://mininet.org/

1.5.1 *Principales características*

Las principales características del emulador Mininet son: (Martínez, 2015, p.34).

- **Flexible**: Mediante el uso de lenguajes de programación python en plataformas como Linux y Windows, su principal utilidad se da con redes definidas por software permitiendo la interpretación de nuevas topologías y funcionalidades.

- **Desplegable**: No exige cambios en el código o en la configuración de un dispositivo final en cuanto a la implementación de un prototipo funcional.

- **Interactivo**: La administración y gestión se realiza en tiempo real, como si se estuviese trabajando sobre una red física.
- **Escalable**: En un solo computador se permite la escalabilidad de miles de switches.
- **Compartible**: La experimentación de prototipos independientes, poder realizar pruebas y compartir estas ideas desarrolladas es muy sencillo y fácil.

1.5.2 *Ventajas*

Se presenta una serie de ventajas al simular una red mediante Mininet. (Henao, 2015a: p.11)

- Es un Proyecto de código abierto y puede ejecutarse en una máquina virtual, laptop, servidor, en la nube, entre otras plataformas.
- Usa la tecnología OpenFlow que permite el reenvió de paquetes personalizable.

- Fácil creación de topologías personalizadas y reutilizables accesibles al usuario, facilitando tareas de ejecución, edición y depuración de redes.
- Permite el uso de aplicaciones instaladas en el host anfitrión como si estuvieran instaladas en los host virtualizados.
- Cuenta con una API en Python que permite crear redes a partir de líneas de código (Scripts) facilitando el compartir y replicar resultados.

1.5.3 *Desventajas*

Así como se presenta grandes ventajas al simular una red, también este emulador presenta varias desventajas, como por ejemplo: (Henao, 2015a: p. 12).

- Los recursos de memoria y procesamiento en un host anfitrión son balanceados entre los switches y host virtualizados.
- El núcleo Linux se usa para todos los host virtuales, esto limita la ejecución de programas que dependen de BSD (Distribución de Software Berkeley), Windows u otros sistemas operativos.
- Para personalizar el ambiente de trabajo de un switch en Mininet es necesario el uso de controladores externos con las características requeridas por el usuario, es decir que los controladores disponibles en Mininet son estáticos al momento de su creación.
- La función NAT está disponible para conectar la red Mininet a redes externas ya que las redes emuladas en Mininet están por defecto separadas de las redes físicas del host anfitrión.
- Mininet opera en base a tiempo real lo que limita algunos resultados en las emulaciones.

CAPITULO II

2 DESARROLLO DE REDES DEFINIDAS POR SOFTWARE.

A continuación se plantea un diagrama de bloques, en el cual se describe la metodología usada para desarrollar la plataforma requerida.

2.1 Diagrama de Bloques de la Metodología

El siguiente diagrama de bloques servirá para guiarse en el proceso del desarrollo de la plataforma a implementar:

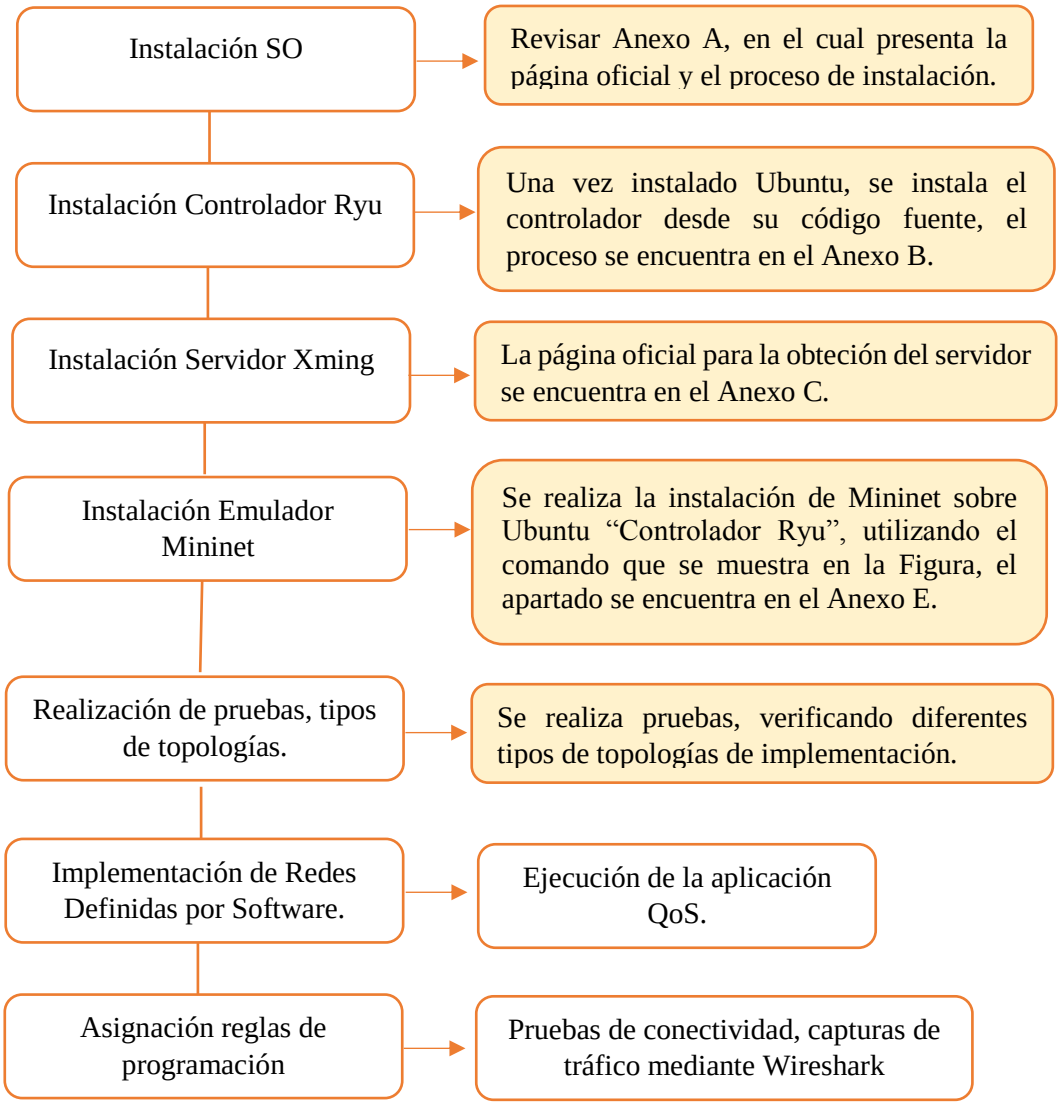


Figura 1-2. Metodología de investigación
Realizado por: GUERRERO, Daniela. 2017

2.2 Instalación herramientas software

Requerimientos para evaluar calidad y servicio de la red a implementar

A continuación, los requerimientos que se presentan en la **Tabla 1-2** ya que sirven para implementar redes SDN, con el objetivo de automatizar reglas de comportamiento.

Tabla 1-2: Requerimientos para evaluar calidad y servicio

Controlador Ryu	Aplicaciones-scripts escritos en Python	rest_firewall.py
Mininet	Emulador de redes	
Wireshark	Captura tráfico	Tráfico ICMP Cambio de prioridad

Realizado por: GUERRERO, Daniela. 2017

2.2.1 *Instalación del SO Ubuntu*

Ubuntu es un sistema operativo muy popular, constantemente actualizado y compatible con todo. Está compuesto por diversos paquetes de software que son distribuidos bajo código abierto y licencia libre. En la página oficial *www.ubuntu.com* se inicia la descarga de la imagen del servidor llamado *ubuntu-14.04.3.* (Ubuntu, 2017. Disponible en: https://www.ubuntu.com/server). El SO se carga en Virtual Box. En el **ANEXO A** se realiza el proceso de instalación.

2.2.2 *Instalación Controlador Ryu*

Ryu es un framework escrito totalmente en lenguaje de programación Python, admite versiones 1.0, 1,1, 1.2, 1.3, 1.4 y 1.5 de OpenFlow. Cuenta con comunidades de gran ayuda como manejo del repositorio de código fuente GitHub, el *Mailing List*, y varios libros, como por ejemplo, en el link web http://osrg.github.io/ryu-book/en/Ryubook.pdf se encuentra disponible uno de los documentos creados por el equipo RYU. Se hace el uso de sus aplicaciones OpenFlow 1.3. La instalación del controlador Ryu realmente es muy sencilla, sin embargo se necesitan algunos componentes muy importantes para su correcto funcionamiento y se muestran en la **Tabla 1-2.**

Tabla 2-2: Componentes necesarios en la preinstalación de Ryu

Componentes	Funcionalidad
apt-get update	Si existen cambios que se aplican a un programa, se utiliza para corregir errores, agregarle funcionalidad, actualizarlo, etc.
apt-get install python-pip	Sirve para instalar y administrar paquetes de software escritos en Python
apt-get install python-dev	Desarrolladores de Python

apt-get install python-eventlet	Es una biblioteca de red simultánea para Python que le permite cambiar la forma en que ejecuta su código, no cómo lo escribe.
apt-get install python-routes	Routes es una reimplementación de Python del sistema de rutas de Rails para mapear URL a Controladores/Acciones y generar URL.
apt-get install python-webob	Proporciona objetos para solicitudes y respuestas HTTP.
apt-get install python-paramiko	En sí misma es una interfaz pura de Python en torno a los conceptos de redes SSH.

Realizado por: GUERRERO, Daniela. 2017
Fuente: www.python.org

Después de la correcta instalación de los componentes antes mencionados, se realiza la instalación final del controlador Ryu y se puede observar en el apartado del **ANEXO B**.

2.2.3 *Instalación del servidor Xming*

El servidor Xming es una interfaz del sistema de ventanas X, es muy útil para configurar de forma independiente cada host virtualizado en Mininet, también permite el acceso al programa Wireshark para realizar capturas de paquetes OpenFlow de la emulación. La descarga se realiza en la página oficial que se muestra en el **ANEXO C**.

2.2.4 *Instalación Emulador Mininet*

Este emulador es una importante herramienta que se desarrolló con la creciente aceptación de las Redes Definidas por Software, cuenta con una API de Python que a partir de un conjunto de líneas de código permite la construcción y manejo de redes de datos. Es así que, Mininet ofrece un entorno de línea de comandos simple que permite una fácil interacción con el usuario. En el **ANEXO D** se muestra el proceso detallado de instalación de este emulador de redes.

2.3 Comandos básicos en la herramienta Mininet

El funcionamiento de una Red Definida por Software requiere de un controlador, una aplicación software que se ejecute en el controlador, un conjunto de equipos de red (swtiches), un software instalado en los switches y una coincidencia entre el protocolo y versión utilizada para la comunicación entre la capa de infraestructura y la de control, como es OpenFlow. Por defecto Mininet adiciona un controlador y un software switch para los procesos de emulación e instala en ellos la versión 1.0 de OpenFlow. Además instala y corre en el controlador una aplicación denominada switch simple. Mininet posibilita la generación de diferentes topologías de red, estas son: minimal, single, linear y tree; varían en número de switches, los enlaces entre éstos y el número de hosts.

2.3.1 *Estructura de comandos para emulación de redes*

El comando principal de Mininet es *sudo mn*, el cual inicia el emulador creando una *topología minimal*, la cual está compuesta por dos hosts conectados a un switch; en cambio la *topología single* también es una topología con un único switch pero la diferencia es la cantidad de hosts que puede indicarse por medio de un parámetro, permitiendo personalizar la topología creada y su funcionamiento por medio de opciones adicionales como se muestra en la comando de la **Figura 2-2** y en la **Tabla 2-2** se detalla las opciones a utilizar según la topología deseada.

```
sudo mn --[OPCIÓN]=[PARÁMETRO],[ARGUMENTOS] …
```

Figura 2-2. Comando general Mininet.
Fuente: www.mininet.org

Tabla 3-2: Estructura de comandos en Mininet

OPCIÓN	PARÁMETRO	ARGUMENTOS
help		
switch	default, ivs, ovsbr, ovsk, ovsl, user, lxbr	lxbr: stp=[1|0] ovsbr: stp=[1|0]
host	cfs,rt	
controller	none, default, nox, pox, ryu, lvs, ovsc	lvs: ip=[IP],port=[PUERTO]
link	default, tc	tc: bw=[BW],delay=[TIME], loss=[%]
topo	linear, minimal, single, reserved, tree, torus	linear: k=[SW],n=[HOST] single: k=[HOST] reserved: k=[HOST] tree:depth=[ALTURA],fanout=[RAMAS] torus: x=[N],t=[N]
clean		
custom	<fichero.py>	
test	cli, none, build, pingpair, pingall, iperf, iperfudp, all	
xterms		
ipbase	[IP]/[MASK]	
mac		
arp		
verbosity	critical, error, warning, info, debug, output	
inamespace		
listenport	[PUERTO]	
nolistenport		
nat		
version		

Realizado por: GUERRERO, Daniela. 2017
Fuente: HENAO, José. 2015b, pg. 37

2.3.2 *Tipos de Topologías*

Existen tres tipos de topologías, los cuales a continuación se realizan pruebas para una mejor práctica y aprendizaje.

Los mecanismos que se utilizan para emular ambiente de trabajo en Mininet son:

1. Generar el comando en Mininet que permita emular el tipo de topología requerido por el usuario.
2. Ejecutar en el controlador Ryu la aplicación que sirva para dar inteligencia a la red implementada.
3. Ejecutar reglas de comportamiento en el switch, mediante el controlador.
4. Verificar pruebas de conectividad, con diferentes tipos de acceso.

2.3.2.1 *Topología Single*

Se genera una topología parecida a la *"topología minimal"*, la diferencia es que este tipo de topología permite conectar más hosts en el mismo switch. El comando de la **Figura 3-2** crea un switch conectado a 7 hosts.

```
sudo mn --topo=single,7 --mac --switch=ovsk,protocols=OpenFlow13
--controller=remote,ip=192.168.1.8
```

Figura 3-2: Comando Topología Single.
Realizado por: GUERRERO, Daniela. 2017

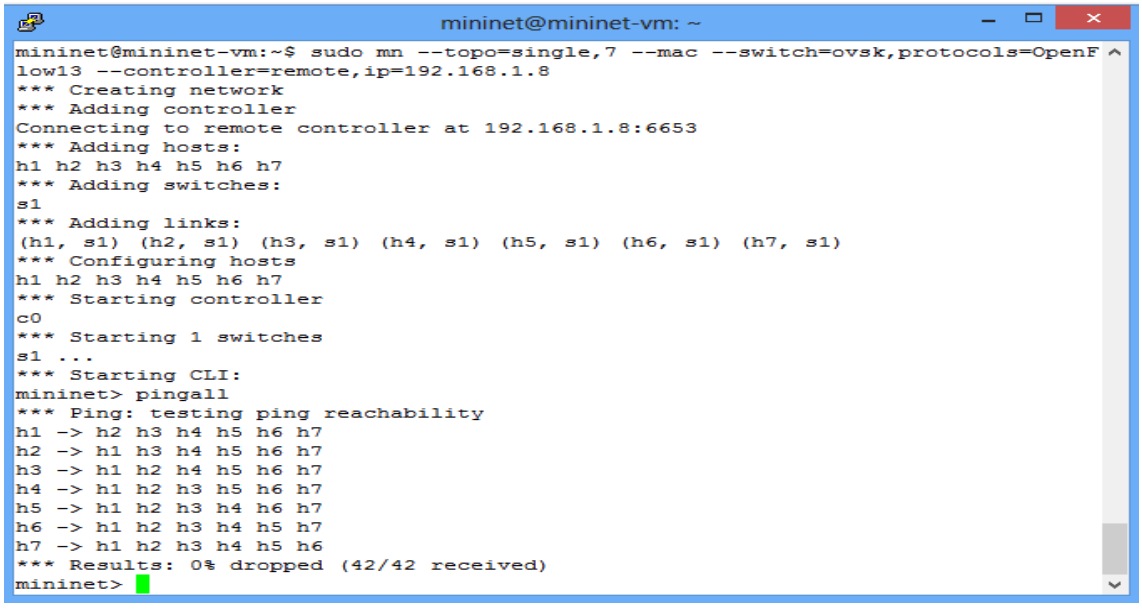


Figura 4-2. Mininet-Topología single
Realizado por: GUERRERO, Daniela. 2017

A continuación en el controlador se usa *simple_switch_13.py* del conjunto de aplicaciones de Ryu para dar inteligencia lógica a la red creada en Mininet y *gui_topology.py* servirá para generar peticiones al servidor Web, permitiendo observar dicha topología en el URL: http://localhost:8080, localhost se refiere a la IP del controlador, el comando de la **Figura 5-2** conecta la red así:

```
ryu-manager gui_topology/gui_topology.py simple_switch_13.py --observe-links
```

Figura 5-2: Comando generador de script switch simple 1.3 en Ryu.
Realizado por: GUERRERO, Daniela. 2017
Fuente: MURILLO, Pablo. 2015, pg. 40

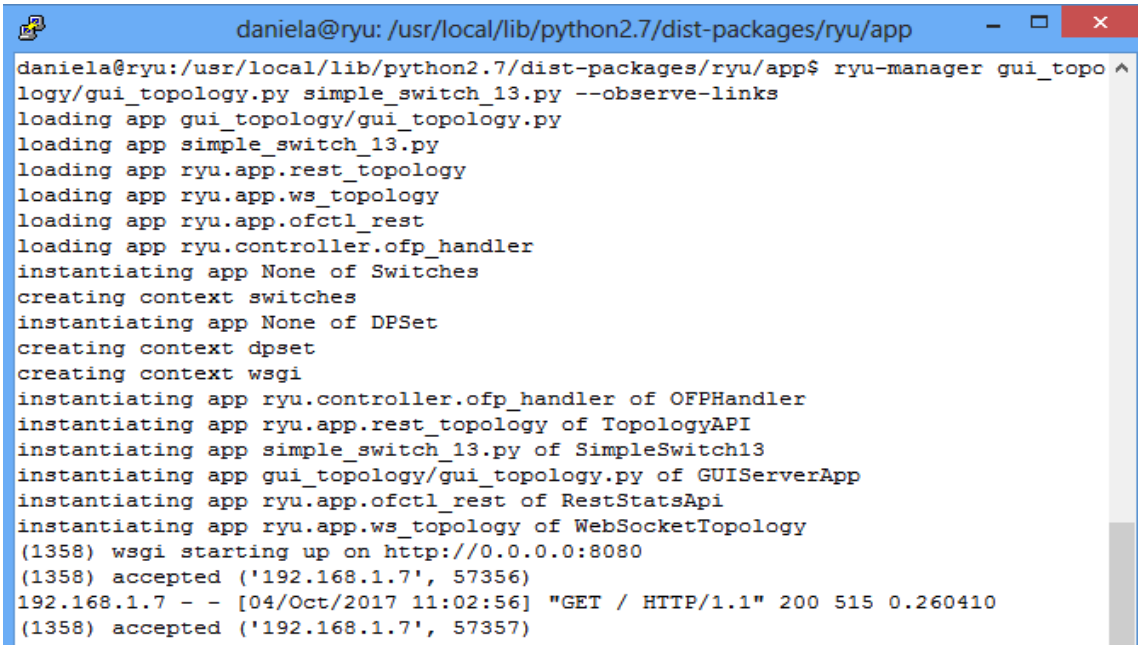


Figura 6-2. Generación switch simple, topología single
Realizado por: GUERRERO, Daniela. 2017

La opción - *-observe-links* que se muestra en la **Figura 6-2**, servirá para poder visualizar la interfaz web de la red mediante la URL: http: //192.168.1.8:8080/, 192.168.1.8 en este caso es la ip del controlador. Se observa que se detalla la conexión exitosa entre el controlador y la interfaz web mediante los componentes wsgi y dpset que se generan internamente, dando como resultado *"accepted"*.

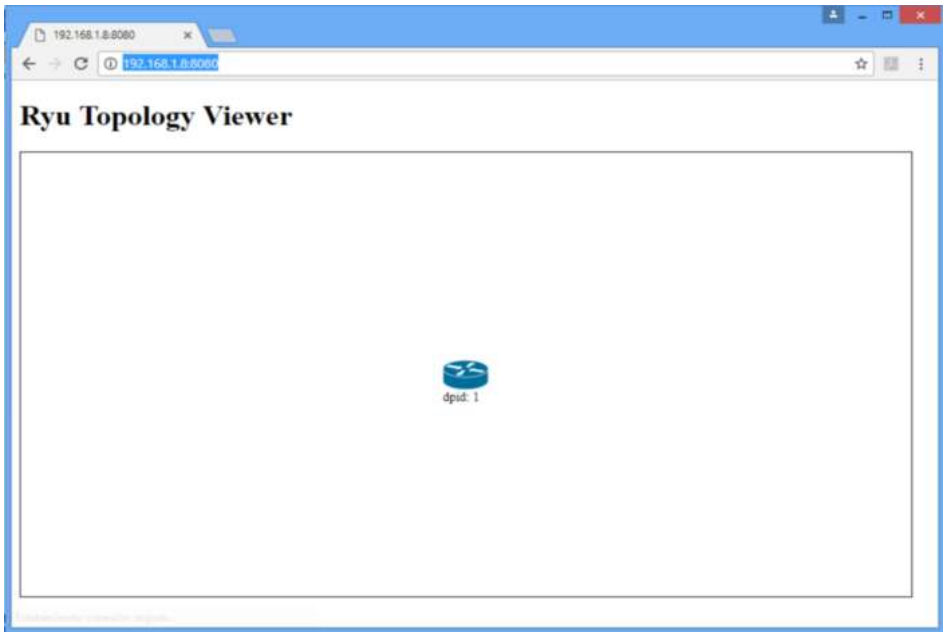


Figura 7-2. Topología Single mediante interfaz web.
Realizado por: GUERRERO, Daniela. 2017

Al realizar pingall en Mininet como se muestra en la **Figura 8-2** se puede observar la conexión exitosa de los cinco hosts creados y que los flujos se transmiten de manera satisfactoria entre ellos y el detalle de la misma se muestra en la **Figura 9-2.**

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7
h2 -> h1 h3 h4 h5 h6 h7
h3 -> h1 h2 h4 h5 h6 h7
h4 -> h1 h2 h3 h5 h6 h7
h5 -> h1 h2 h3 h4 h6 h7
h6 -> h1 h2 h3 h4 h5 h7
h7 -> h1 h2 h3 h4 h5 h6
*** Results: 0% dropped (42/42 received)
mininet>
```

Figura 8-2. Comando pingall_conexión exitosa de los hosts.
Realizado por: GUERRERO, Daniela. 2017

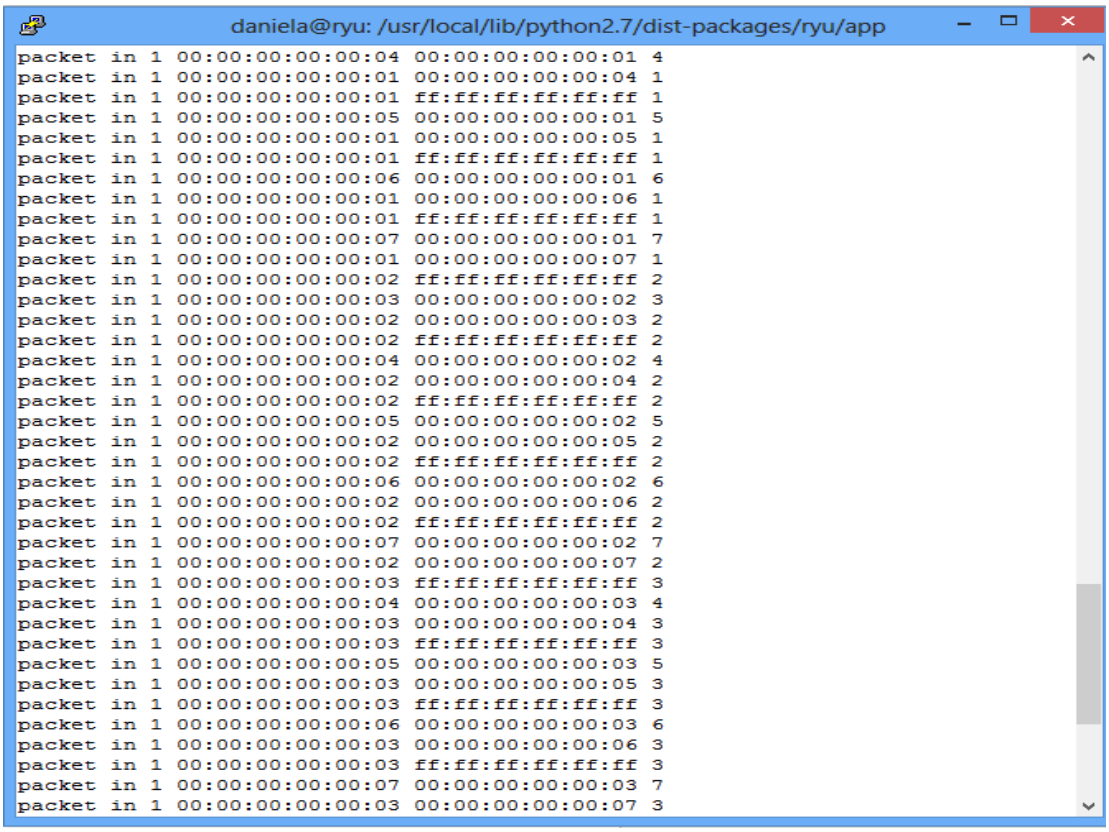


Figura 9-2. Detalle de la transmisión de los flujos entre hosts.
Realizado por: GUERRERO, Daniela. 2017

Los hosts no se observan en la topología de la **Figura 7-2**, por lo que se puede utilizar la herramienta xming, para abrir ventanas externas de los hosts.

2.3.2.2 *Topología Linear*

Se crea una topología Linear con el comando de la **Figura 10-2**, se usa la IP del controlador Ryu 192.168.1.8 y se especifica la versión OpenFlow 1.3 ya que en el controlador existen aplicaciones que trabajan con diferentes versiones.

```
sudo mn --mac --topo=linear,2 --switch=ovsk,protocols=OpenFlow13
--controller=remote,ip=192.168.1.8
```

Figura 10-2: Comando topología Linear.
Realizado por: GUERRERO, Daniela. 2017

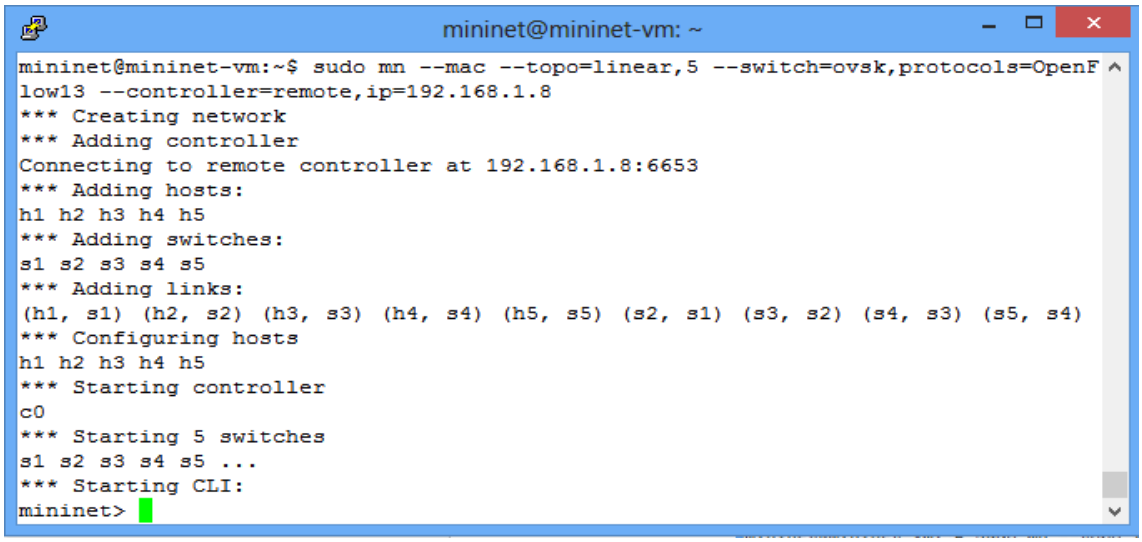


Figura 11-2. Mininet-Topología Linear
Realizado por: GUERRERO, Daniela. 2017

Figura 12-2. Generación switch simple, topología linear
Realizado por: GUERRERO, Daniela. 2017

Los hosts no se observan en la topología de la **Figura 13-2**, por lo que se puede utilizar la herramienta xming, para abrir ventanas externas de los hosts.

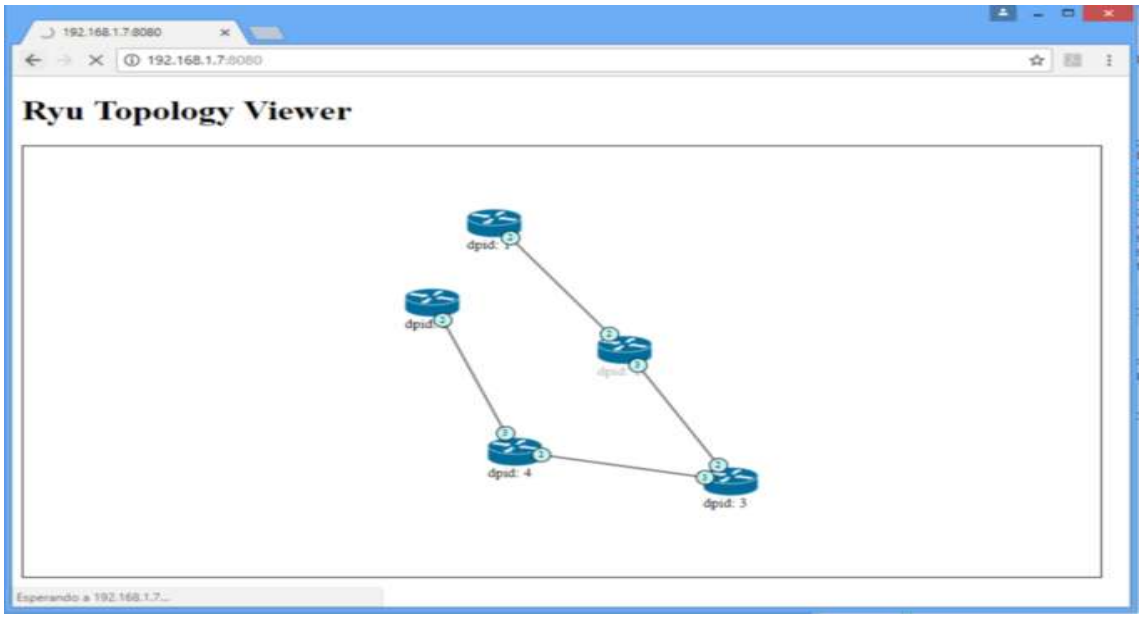


Figura 13-2. Topología Linear mediante interfaz web.
Realizado por: GUERRERO, Daniela. 2017

2.3.2.3 *Topología Tree (Árbol)*

Ahora se prueba una topología en la que un switch se conecta a tres switches y cada uno estos a tres switches simultáneamente. Se ejecuta el comando de la **Figura 14-2** para diseñar dicha topología y visualizarla en el controlador Ryu mediante la Interfaz Web.

```
sudo mn --topo=tree,fanout=3,depth=3 --mac --switch=ovsk,protocols=OpenFlow13 --controller=remote,ip=192.168.1.7
```

Figura 14-2: Comando Topología Árbol.
Realizado por: GUERRERO, Daniela. 2017

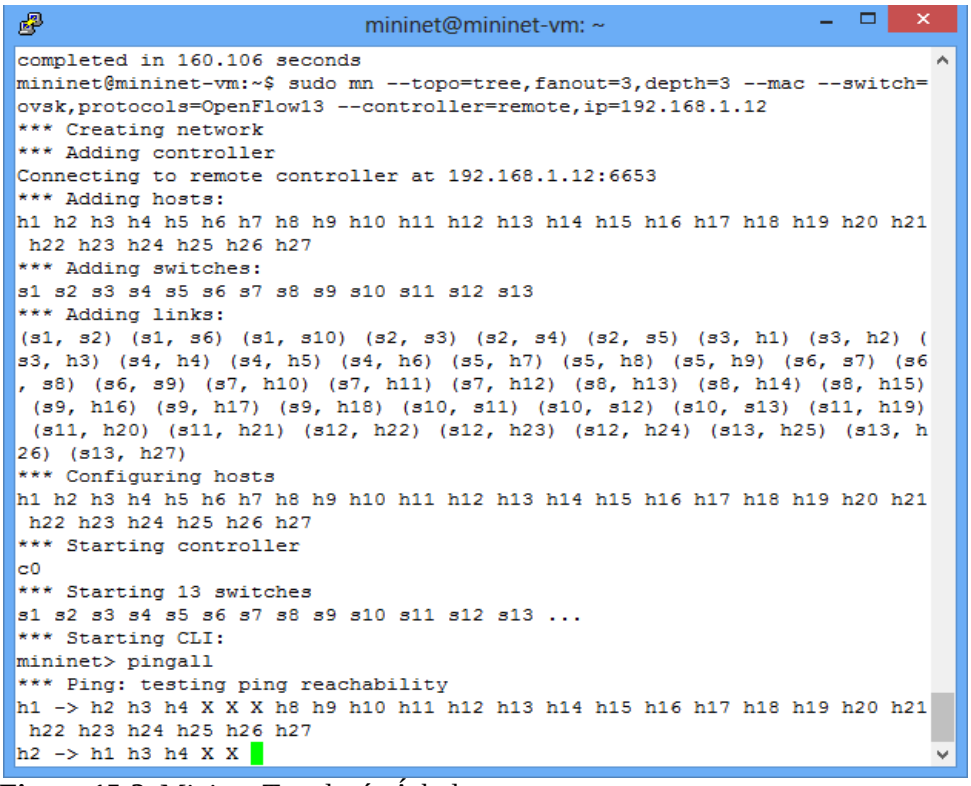


Figura 15-2. Mininet-Topología Árbol
Realizado por: GUERRERO, Daniela. 2017

En el controlador se ejecuta nuevamente como en el ejemplo anterior, la aplicación *gui_topology.py* para generar una interfaz web y se pueda observar los dispositivos virtuales creados. Despues de su ejecución, se observa los switches creados como se observa en la **Figura 17-2**.

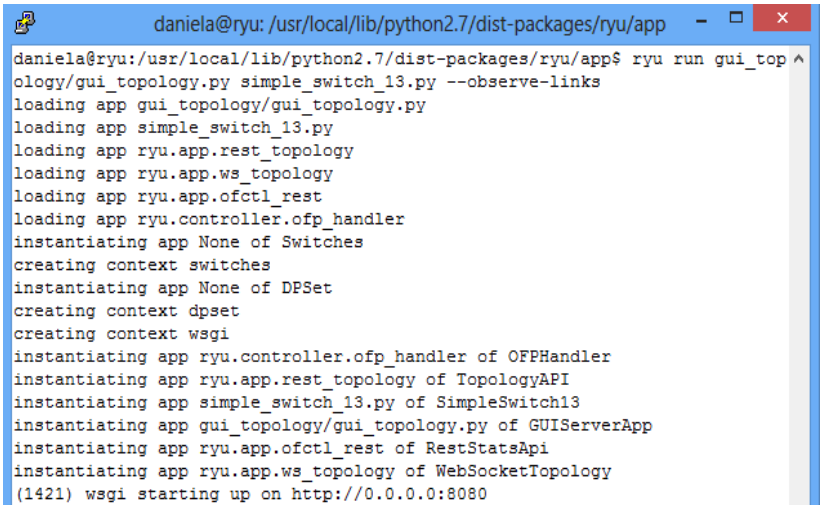


Figura 16-2. Generación switch simple, topología árbol.
Realizado por: GUERRERO, Daniela. 2017

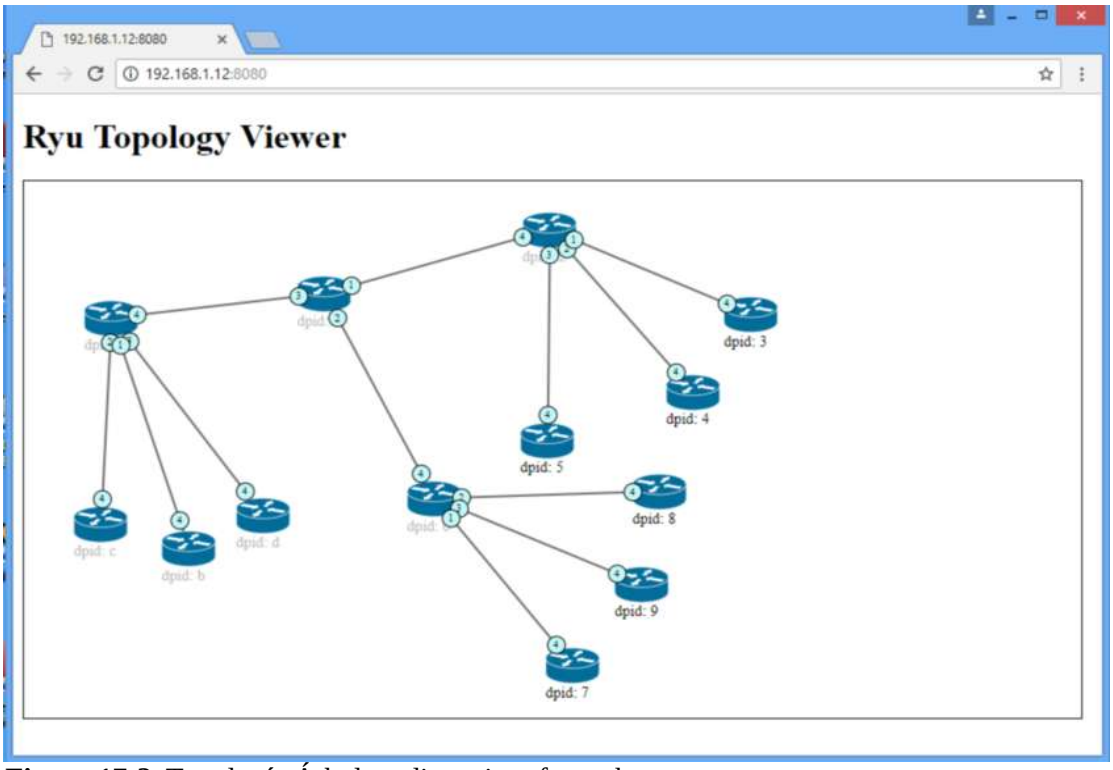


Figura 17-2. Topología Árbol mediante interfaz web
Realizado por: GUERRERO, Daniela. 2017

2.4 Implementación Red Definida por Software

2.4.1 *Diseño de Red*

El diseño de red de este proyecto se basa básicamente en diseñar cinco hosts conectados a un mismo switch, la **Figura 18-2** muestra la forma final de la red emulada. Se utiliza el controlador Ryu para insertar la inteligencia lógica a la red, el controlador genera un ID: 0000000000000001 al switch, este sirve para conectar y almacenar las reglas que más adelante se detallarán sus funcionalidades.

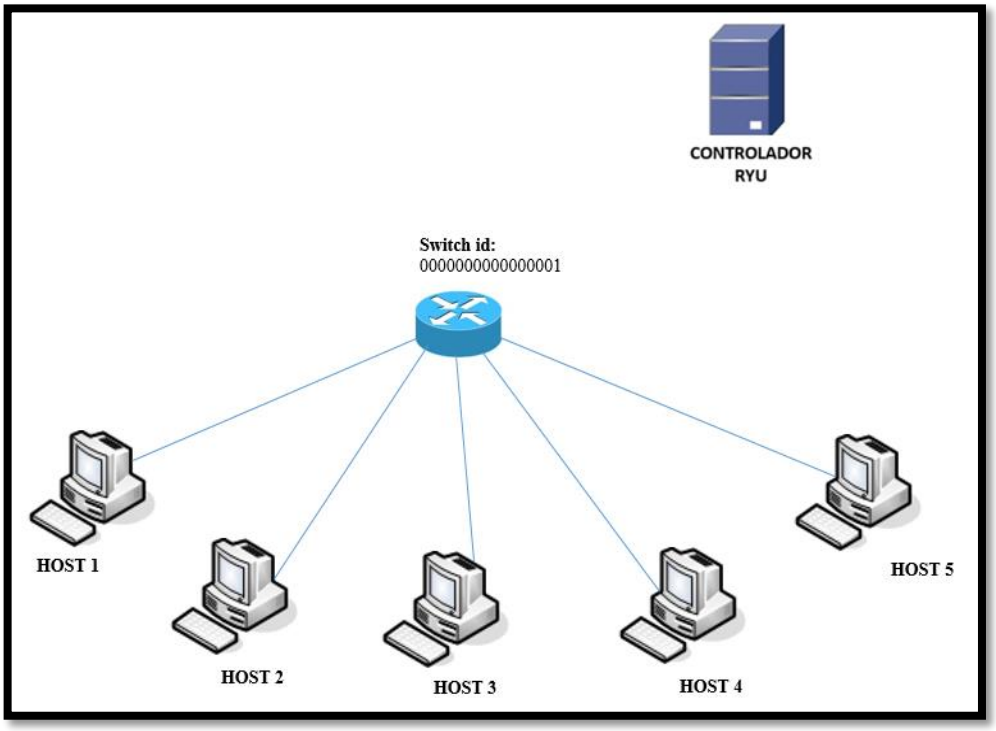


Figura 18-2. Topología de red diseñada
Realizado por: GUERRERO, Daniela. 2017

```
sudo mn --mac --topo=single,5 --switch=ovsk,protocols=OpenFlow13 --
controller=remote,ip=192.168.1.8
```

Figura 19-2. Topología Single con cinco hosts conectados.
Realizado por: GUERRERO, Daniela. 2017

Cuando se ejecuta en Mininet el comando de la **Figura 19-2**, se crea automáticamente el controlador el switch y cinco hosts conectados a este. Pero esta red no es capaz de transmitir datos entre hosts porque no tiene ninguna aplicación corriendo en la red, es por esto que se utiliza el controlador.

Figura 20-2. Diseño de red, topología single con cinco hosts.
Realizado por: GUERRERO, Daniela. 2017

2.4.2 *Aplicación sobre el Controlador Ryu*

A continuación se duplica la sesión putty del controlador Ryu, la primera ventana Ryu servirá para generar la aplicación y la segunda ventana de Ryu servirá para asignar reglas de comportamiento de la red. En la **Figura 21-2** se genera el código fuente de la aplicación QoS del repositorio de Ryu. La aplicación en Ryu es llamado rest_firewall.py, el cual permitirá el cambio de prioridad de paquetes entre los hosts, así como también permitir o bloquear tráfico ICMP.

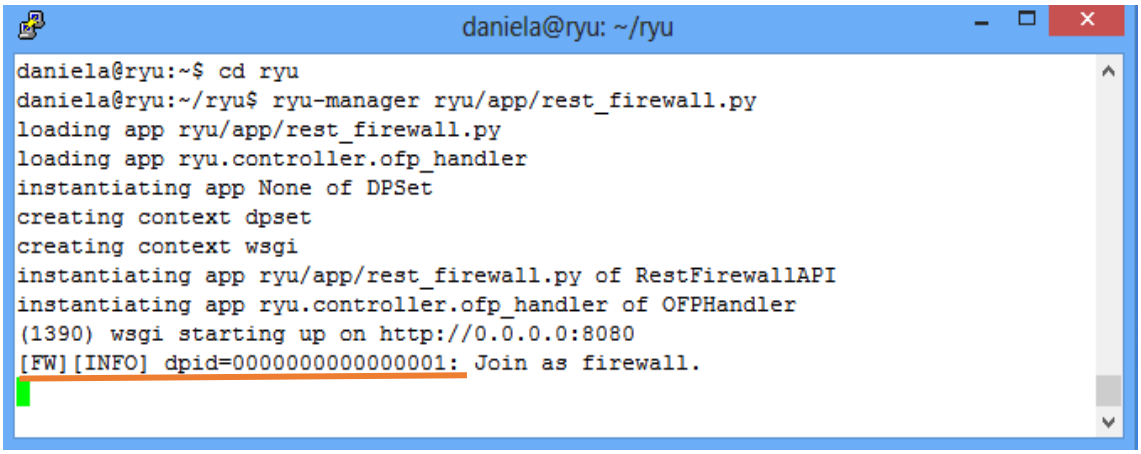


Figura 21-2. Conexión exitosa entre el switch y el controlador
Realizado por: GUERRERO, Daniela. 2017

Se verifica al final de la ejecución la conexión satisfactoria de la aplicación entre el switch y el controlador Ryu, en la **Figura 22-2** se muestra como fue la conexión**.**

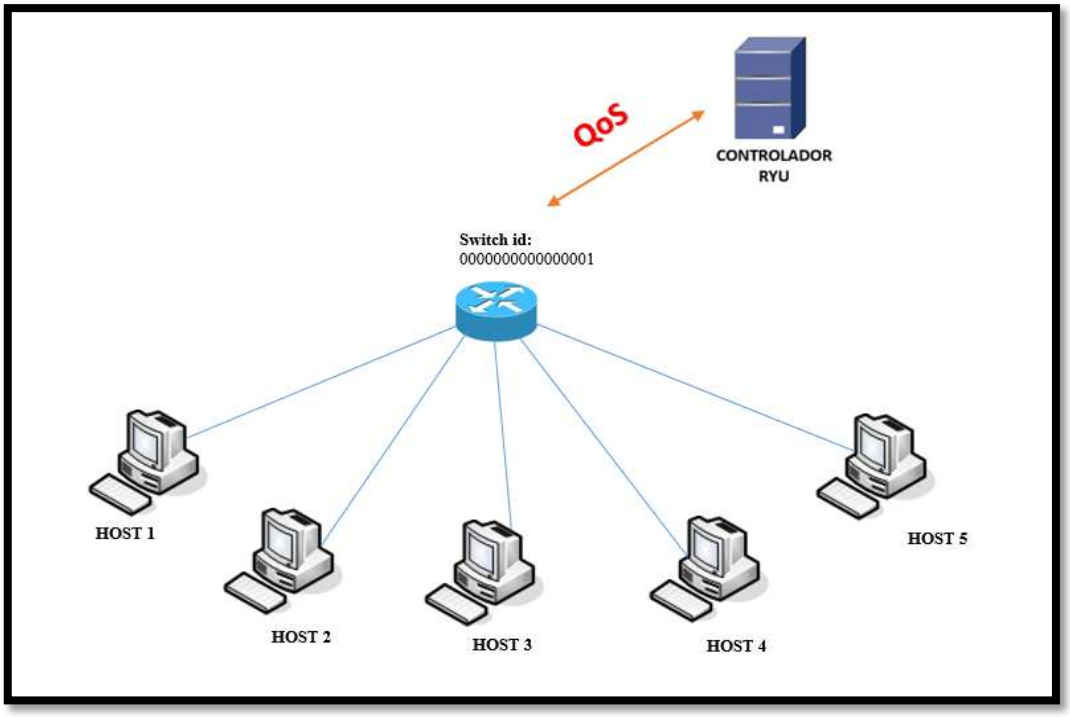


Figura 22-2. Gráfico de conexión entre el switch y controlador mediante aplicación
Realizado por: GUERRERO, Daniela. 2017

2.4.2.1 *Lista REST API*

La red virtual ahora es controlada mediante la aplicación de QoS, esta aplicación toma el control de los flujos de paquetes que se transmiten entre los dispositivos virtuales creados, permitiendo o denegando el acceso a protocolos de comunicación, para lograr este objetivo se requiere tener conocimiento previo sobre los métodos, estructuras y observaciones de las reglas que se van aplicar en el controlador Ryu. Estas son:

- Ver el estado de todos los switches
- Cambiar el estado de todos los switches
- Adquirir información de todas las reglas
- Añadir reglas
- Eliminar reglas
- Adquirir el estado de salida de log de todos los switches

Se detalla a continuación la estructura de cada una de las reglas:

1. VER EL ESTADO DE TODOS LOS SWITCHES

MÉTODO: GET
URL: http://localhost:8080/firewall/module/status

EJEMPLO:

```
daniela@ryu:~$ curl http://localhost:8080/firewall/module/status
```

Figura 23-2. Comando_ver estado del switch
Realizado por: GUERRERO, Daniela. 2017

```
[{"status": "enable", "switch_id": "0000000000000001"}]daniela@ryu:~$
```

Figura 24-2. Resultado_estado del switch
Realizado por: GUERRERO, Daniela. 2017

2. CAMBIAR EL ESTADO DE TODOS LOS SWITCHES

El estado inicial de cada switch creado es "disable", por lo que al inciar con la ejecución se debe cambiar a estado "enable".

MÉTODO: PUT
URL: http://localhost:8080/firewall/module/{op}/{switch}
Op: ["enable"/"disable"]
Switch: ["all"/ID_SWITCH]

EJEMPLO:

```
daniela@ryu:~$ curl -X PUT http://localhost:8080/firewall/module/enable/0000000000000001
```

Figura 25-2. Comando_cambiar estado del switch
Realizado por: GUERRERO, Daniela. 2017

```
[{"switch_id": "0000000000000001", "command_result": {"result": "success", "details": "fi
rewall running."}}]daniela@ryu:~$
```

Figura 26-2. Resultado_estado enable del switch
Realizado por: GUERRERO, Daniela. 2017

3. ADQUIRIR INFORMACIÓN DE TODAS LAS REGLAS

La especificación de la identificación Vlan es opcional.

MÉTODO: GET
URL: http://localhost:8080/firewall/rules/{switch}/{vlan}
switch: ["all"/switch _ID]
vlan: ["all"/vlan_ID]

EJEMPLO:

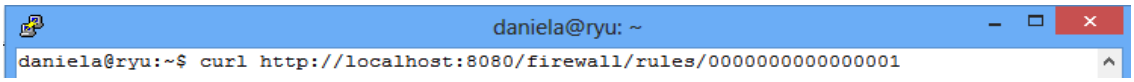


Figura 27-2. Comando_información de reglas añadidas
Realizado por: GUERRERO, Daniela. 2017

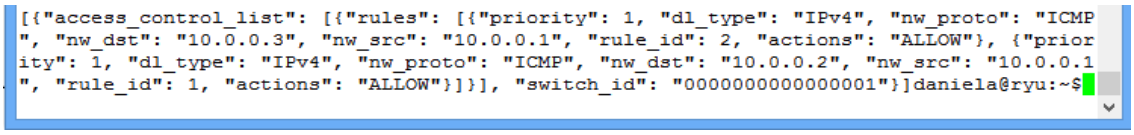


Figura 28-2. Resultado_ información de reglas añadidas
Realizado por: GUERRERO, Daniela. 2017

4. AÑADIR REGLAS

Cuando el registro de reglas es satisfactorio, se genera y se guarda la ID de cada regla.

MÉTODO: POST
URL: http://localhost:8080/firewall/{switch}/{vlan}
switch: ["all"/switch _ID]
vlan: ["all"/vlan_ID]
DATOS:
Prioridad: [0-65535]
Puerto: [0-65535]
dl-src: "xx:xx:xx:xx:xx:xx"
dl-dst: "xx:xx:xx:xx:xx:xx"
dl_type: [ARP/IPv4/IPv6]
nw_src: "xxx:xxx:xxx:xxx/xx"
nw_dst: "xxx:xxx:xxx:xxx/xx"
ipv6_src: "xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx/xx"
nw_proto: [TCP/UDP/ICMP/ICMPv6]
tp_src: [0-65535]
tp_dst: [0-65535]
actions: [ALLOW/DENY]

EJEMPLO:

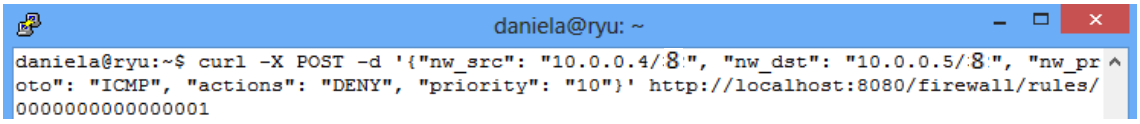


Figura 29-2. Comando_añadir regla
Realizado por: GUERRERO, Daniela. 2017

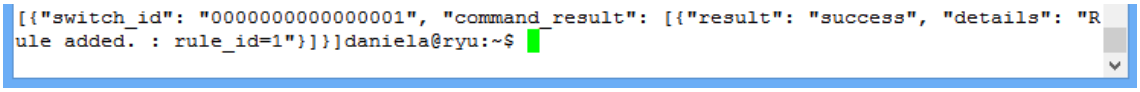


Figura 30-2. Resultado_añadir regla
Realizado por: GUERRERO, Daniela. 2017

5. ELIMINAR REGLAS

La especificación de la identificación Vlan es opcional.

MÉTODO: DELETE
URL: http://localhost:8080/firewall/rules/{switch}/{vlan}
switch: ["all"/switch _ID]
vlan: ["all"/vlan_ID]
DATOS: rule_id: ["all"/1-..]

EJEMPLO:

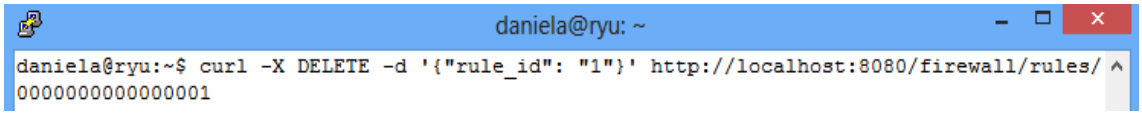


Figura 31-2. Comando_eliminar regla
Realizado por: GUERRERO, Daniela. 2017

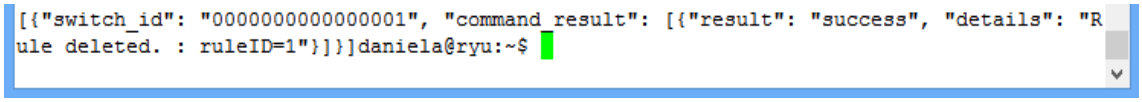


Figura 32-2. Resultado _eliminar regla
Realizado por: GUERRERO, Daniela. 2017

6. ADQUIRIR EL ESTADO DE SALIDA DE LOG DE TODOS LOS SWITCHES

MÉTODO: GET
URL: http://localhost:8080/firewall/log/status

EJEMPLO:

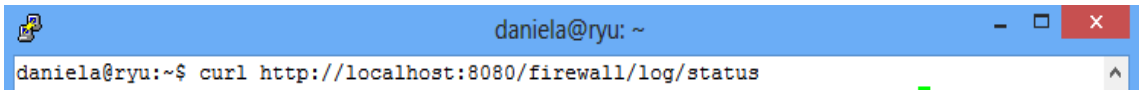


Figura 33-2. Comando _estado de salida de log del switch
Realizado por: GUERRERO, Daniela. 2017

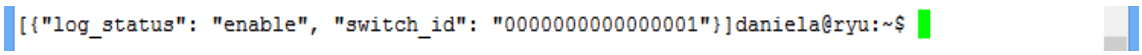


Figura 34-2. Resultado_estado de salida de log del switch
Realizado por: GUERRERO, Daniela. 2017

2.5 Asignación de reglas de programación en el controlador Ryu

Se conoció la información necesaria para aplicar reglas de comportamiento de flujos, estas reglas se añaden y se almacenan en el controlador Ryu. El objetivo de este proyecto se enfoca en permitir o bloquear tráfico en la red y cambiar el nivel de prioridad de los flujos, en la **Figura 35-2** se observa claramente el objetivo de este capítulo.

Para lograr esto se duplica la sesión del controlador Ryu, esta ventana servirá para manipular la red con las reglas antes mencionadas.

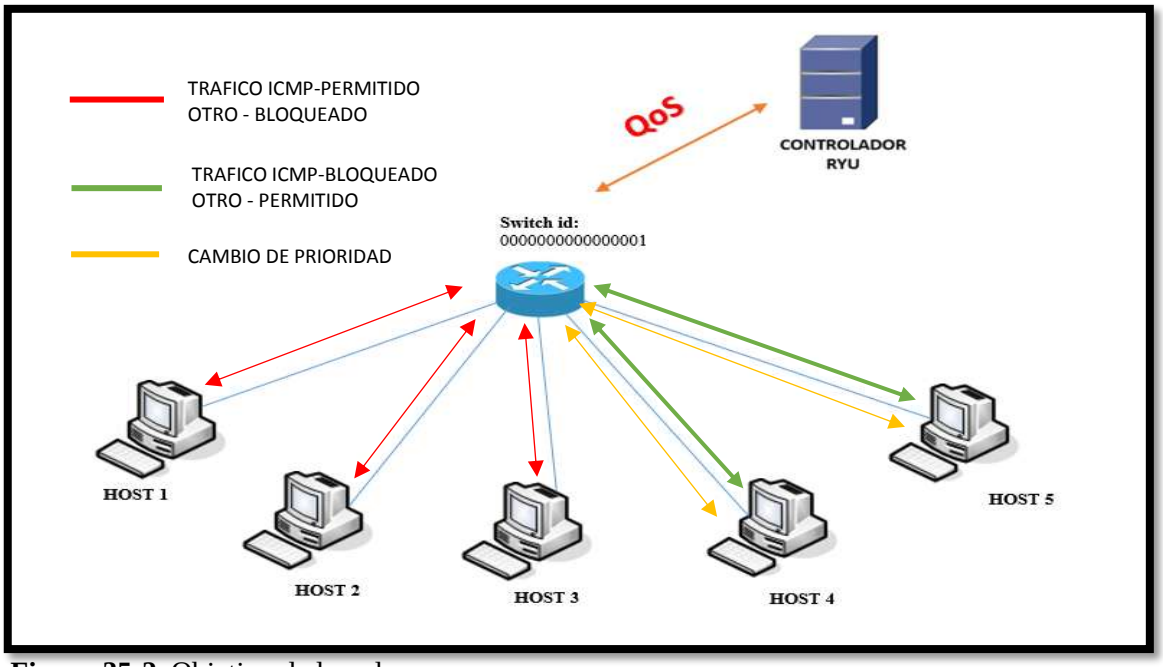


Figura 35-2. Objetivo de la red
Realizado por: GUERRERO, Daniela. 2017

Por defecto, la aplicación que se va aplicar en el controlador esta en estado *"disable"*. A continuación se va hacer uso de la REGLA 2, ya que permite poner en estado *"enable"*. En la **Figura 36-2** se ingresa la regla de comando.

Figura 36-2. Estado enable de la red
Realizado por: GUERRERO, Daniela. 2017

Figura 37-2. Resultado_estado enable de la red
Realizado por: GUERRERO, Daniela. 2017

Al final de la ejecución se refleja el resultado ***"success"*** de la **Figura 36-2**. El resultado ha sido formateado para mejorar la visualización como se indica en la **Figura 37-2.** A continuación en la siguiente **Figura 38-2** se observa que en la primera ventana de Ryu donde se estableció la aplicación, al final de la ejecución se encuentra la regla asignada mostrando en estado *"enable"*.

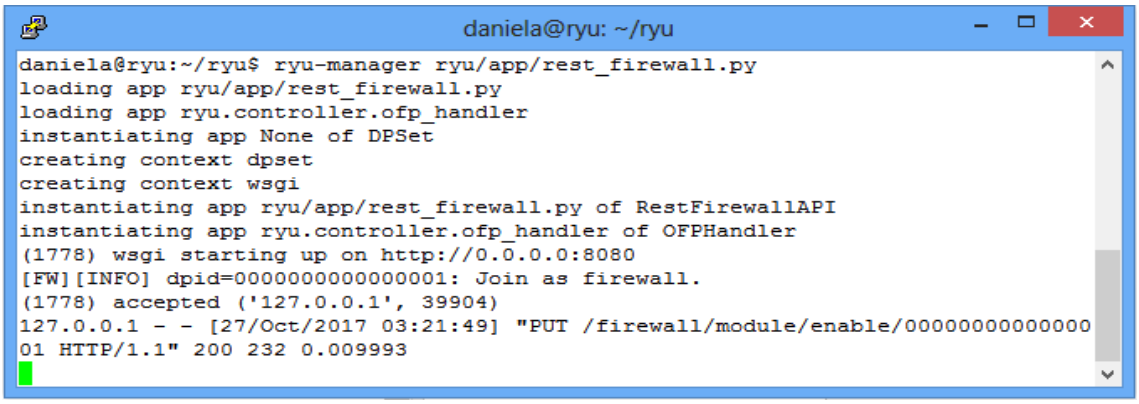


Figura 38-2. Controlador_regla asignada_estado enable
Realizado por: GUERRERO, Daniela. 2017

Se procede a añadir reglas de comportamiento de flujos de paquetes entre los hosts. Mininet crea automáticamente direcciones IP a los hosts. En la **Tabla 4-2** se observa el detalle de las direcciones asignadas automáticamente a los dispositivos virtuales.

Tabla 4-2: Direcciones IP asignadas automáticamente a cada host

NUMERO DE HOST	DIRECCION IP
Host 1	10.0.0.1/8
Host 2	10.0.0.2/8
Host 3	10.0.0.3/8
Host 4	10.0.0.4/8
Host 5	10.0.0.5/8

Realizado por: GUERRERO, Daniela. 2017

2.5.1 *Añadir reglas mediante POST*

2.5.1.1 *Regla 1_Mediante POST*

Se va a permitir transitar tráfico ICMP entre: host 1, host 2 y host 3, esta regla debe ser agregada de ida y vuelta para cada host, el ID de cada regla se asigna automáticamente de acuerdo a como se vayan agregando. El comando de la **Figura 39-2** se utiliza la dirección IP de origen, dirección IP de destino y tipo de tráfico en este caso es ICMP, si no se especifíca el tipo de permiso la regla

por defecto asume que esta permitiendo el paso del tráfico asignado. En la **Tabla 5-2** detallada a continuación. En la **Figura 40-2** se observa el resultado de la regla añadida en el host 1, al igual que en el host 2 y host 3.

```
# curl -X POST -d '{"nw_src": "10.0.0.1/8", "nw_dst": "10.0.0.2/8", "nw_proto": "ICMP"}' http://localhost:8080/firewall/rules/0000000000000001
```

Figura 39-2. Comando_regla 1_tráfico ICMP
Realizado por: GUERRERO, Daniela. 2017

Tabla 5-2: Reglas añadidas_Tráfico ICMP

ORIGEN	DESTINO	PROTOCOLO	PERMISO	REGLA ID	PRIORIDAD
10.0.0.1/8	10.0.0.2/8	ICMP	Allow	1	1
10.0.0.1/8	10.0.0.3/8	ICMP	Allow	2	1
10.0.0.2/8	10.0.0.3/8	ICMP	Allow	3	1
10.0.0.2/8	10.0.0.1/8	ICMP	Allow	4	1
10.0.0.3/8	10.0.0.1/8	ICMP	Allow	5	1
10.0.0.3/8	10.0.0.2/8	ICMP	Allow	6	1

Realizado por: GUERRERO, Daniela. 2017

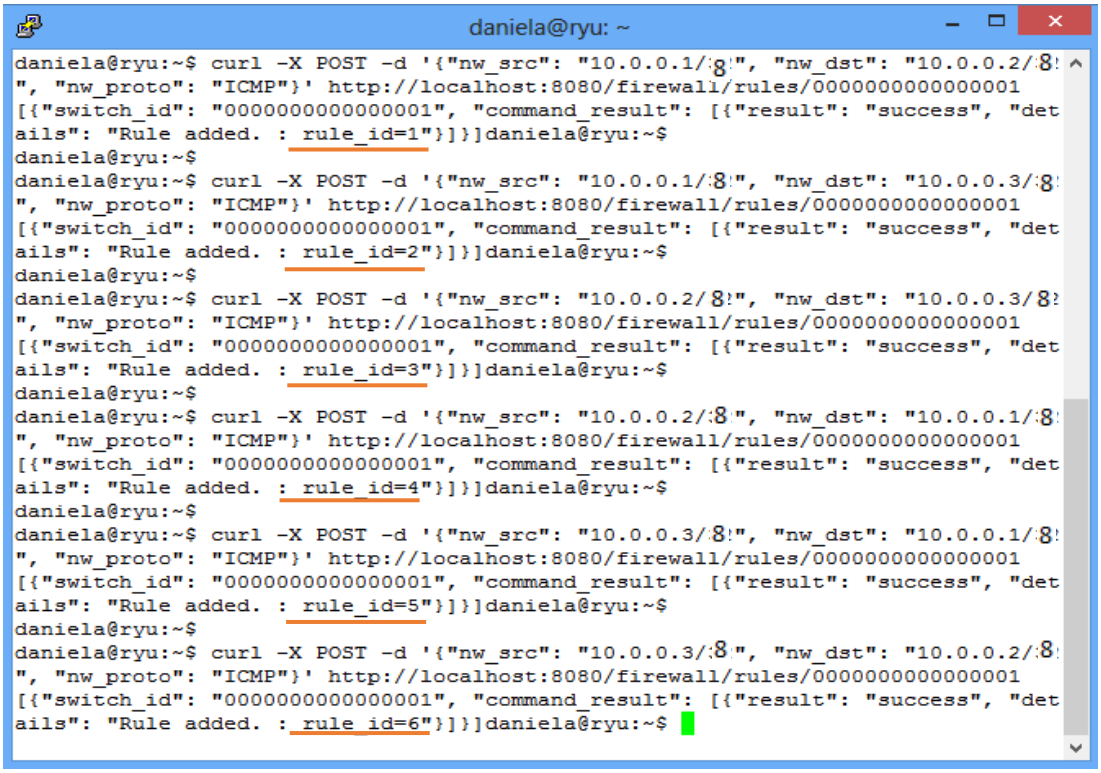


Figura 40-2. Reglas añadidas_Tráfico ICMP
Realizado por: GUERRERO, Daniela. 2017

2.5.1.2 *Regla 2_Mediante POST*

Se adiciona 2 reglas más que van a permitir cualquier tipo de tráfico entre el host 4 y host 5. Igualmente como en las reglas anteriores, se asignan las reglas en cada host de origen y destino.

Tabla 6-2: Reglas añadidas_Cualquier tipo de Tráfico

ORIGEN	DESTINO	PROTOCOLO	PERMISO	REGLA ID	PRIORIDAD
10.0.0.4/8	10.0.0.5/8	any	Allow	7	1
10.0.0.5/8	10.0.0.4/8	any	Allow	8	1

Realizado por: GUERRERO, Daniela. 2017

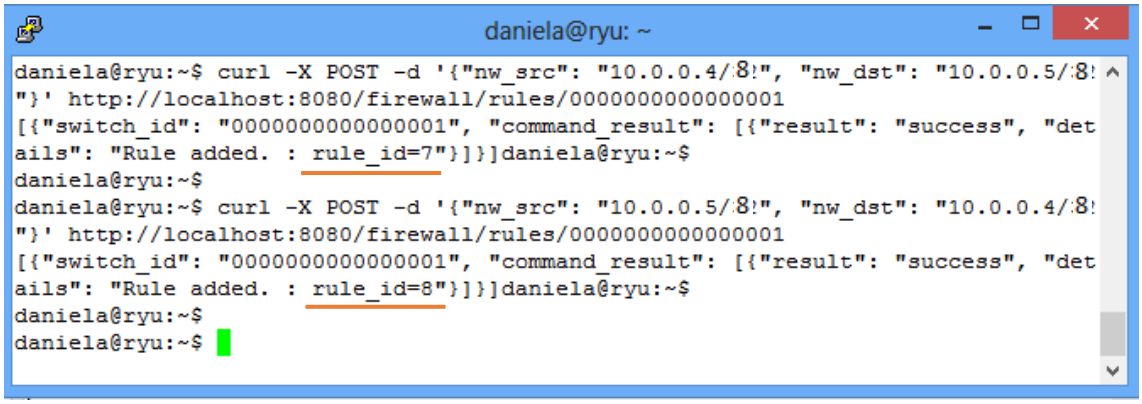


Figura 41-2. Reglas añadidas_Cualquier tipo de Tráfico
Realizado por: GUERRERO, Daniela. 2017

2.5.1.3 *Regla 3_Mediante POST*

Se va a bloquear tráfico ICMP entre el host 2 y host 4, y se cambia el nivel de prioridad de los flujos con valor 10, por defecto el nivel de prioridad es 1.

Tabla 7-2: Reglas añadidas_Bloqueo Tráfico ICMP

ORIGEN	DESTINO	PROTOCOLO	PERMISO	REGLA ID	PRIORIDAD
10.0.0.4/8	10.0.0.5/8	ICMP	DENY	9	10
10.0.0.5/8	10.0.0.4/8	ICMP	DENY	10	10

Realizado por: GUERRERO, Daniela. 2017

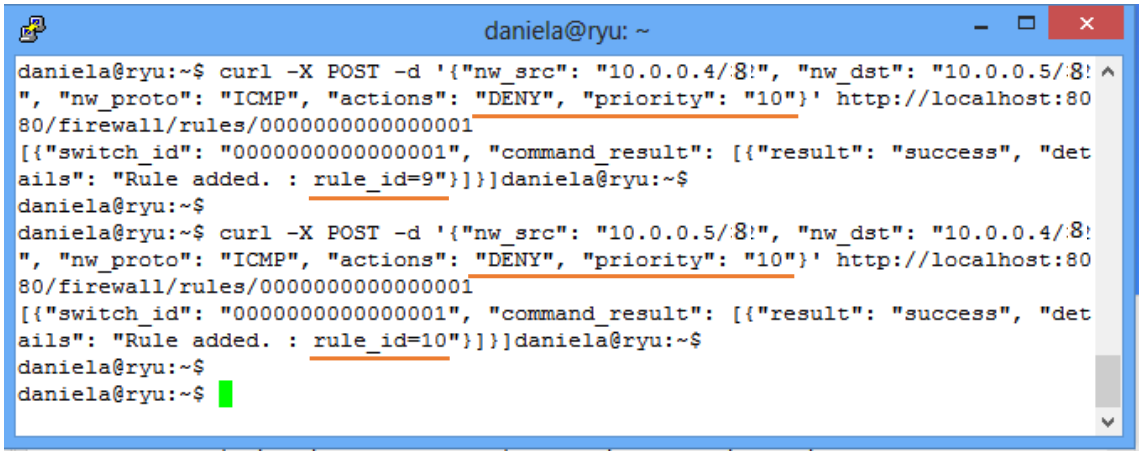


Figura 42-2. Reglas añadidas Bloqueo Tráfico ICMP
Realizado por: GUERRERO, Daniela. 2017

2.6 Registro de reglas

El resultado de todas las reglas ingresadas en el controlador y registradas en el switch se muestra en la **Figura 44-2** y se ha formateado el resultado en la **Tabla** 8-**2** para mejorar la visualización y comprensión del mismo. El comando de la **Figura 43-2** permite mostrar el registro exitoso de todas las reglas.

```
# curl http://localhost:8080/firewall/rules/0000000000000001
```

Figura 43-2. Comando ver todas las reglas añadidas
Realizado por: GUERRERO, Daniela. 2017

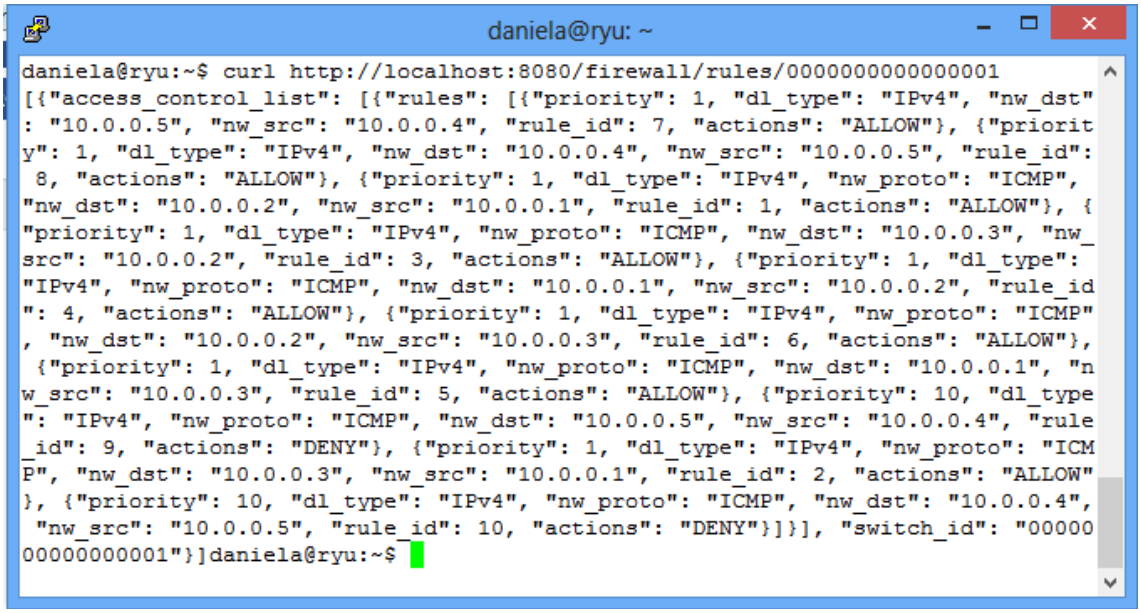


Figura 44-2. Ver todas las reglas añadidas
Realizado por: GUERRERO, Daniela. 2017

Tabla 8-2: Resultado general_todas las reglas añadidas


```
[
  {
    "access_control_list": [
      {
        "rules": [
          {
              "priority": 1,
              "dl_type": "IPv4",
              "nw_dst": "10.0.0.5",
              "nw_src": "10.0.0.4",
              "rule_id": 7,
              "actions": "ALLOW"
          },
          {
              "priority": 1,
```

```
            "dl_type": "IPv4",
            "nw_dst": "10.0.0.4",
            "nw_src": "10.0.0.5",
            "rule_id": 8,
            "actions": "ALLOW"
        },
        {
            "priority": 1,
            "dl_type": "IPv4",
            "nw_proto": "ICMP",
            "nw_dst": "10.0.0.2",
            "nw_src": "10.0.0.1",
            "rule_id": 1,
            "actions": "ALLOW"
        },
        {
            "priority": 1,
            "dl_type": "IPv4",
            "nw_proto": "ICMP",
            "nw_dst": "10.0.0.3",
            "nw_src": "10.0.0.2",
            "rule_id": 3,
            "actions": "ALLOW"
        },
        {
            "priority": 1,
            "dl_type": "IPv4",
            "nw_proto": "ICMP",
            "nw_dst": "10.0.0.1",
            "nw_src": "10.0.0.2",
            "rule_id": 4,
            "actions": "ALLOW"
        },
        {
            "priority": 1,
            "dl_type": "IPv4",
            "nw_proto": "ICMP",
            "nw_dst": "10.0.0.2",
            "nw_src": "10.0.0.3",
            "rule_id": 6,
            "actions": "ALLOW"
        },
        {
            "priority": 1,
            "dl_type": "IPv4",
            "nw_proto": "ICMP",
            "nw_dst": "10.0.0.1",
            "nw_src": "10.0.0.3",
            "rule_id": 5,
            "actions": "ALLOW"
        },
        {
            "priority": 10,
            "dl_type": "IPv4",
```

```
                "nw_proto": "ICMP",
                "nw_dst": "10.0.0.5",
                "nw_src": "10.0.0.4",
                "rule_id": 9,
                "actions": "DENY"
              },
             {
                "priority": 1,
                "dl_type": "IPv4",
                "nw_proto": "ICMP",
                 "nw_dst": "10.0.0.3",
                 "nw_src": "10.0.0.1",
                 "rule_id": 2,
                 "actions": "ALLOW"
               },
              {
                 "priority": 10,
                 "dl_type": "IPv4",
                 "nw_proto": "ICMP",
                 "nw_dst": "10.0.0.4",
                 "nw_src": "10.0.0.5",
                 "rule_id": 10,
                 "actions": "DENY"
              }
             ]
           }
         ],
"switch_id": "0000000000000001"
    }
]
```


Realizado por: GUERRERO, Daniela. 2017

CAPÍTULO III

3 RESULTADOS

3.1 Pruebas de conectividad

Se realiza pruebas de conectividad según las reglas de comportamiento añadidas a la red, para lograr esto se abren ventanas xterm de cada host como se indica en la **Figura 1-3.** También se duplica la sesión en mininet como se muestra en la **Figura 2-3** para acceder a wireshark, el cual permitirá capturar el tráfico de las pruebas a realizar.

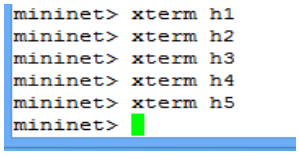


Figura 1-3. Ejecución Ventanas Xterm de cada host
Realizado por: GUERRERO, Daniela. 2017

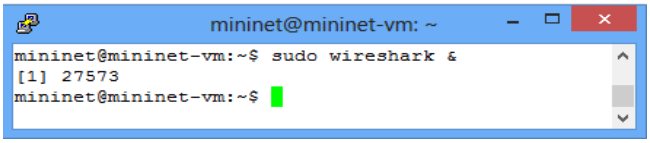


Figura 2-3. Ejecución Wireshark
Realizado por: GUERRERO, Daniela. 2017

3.2 Prueba de conectividad usando diferentes tipos de acceso

3.2.1 *Pruebas de Ping entre los Hosts: 1, 2 y 3.*

En el diseño de la red se permite el paso de tráfico ICMP entre los hosts: 1, 2 y 3 y se bloquea el cualquier otro tipo de tráfico entre ellos, sin cambiar su nivel de prioridad de flujos, el cual por defecto es 1. En cambio en los dos hosts restantes: 4 y 5, se bloquea el tráfico ICMP y se permite cualquier otro tipo de tráfico, cambiando el nivel de prioridad de los flujos a valor 10. Se realiza pruebas de ping entre los todos los hosts para comprobar la eficiencia de la red.

3.2.1.1 *Prueba de Ping entre host 1 y host 2*

Al hacer ping a la dirección de destino 10.0.0.2 del host 2, se comprueba en la **Figura 3-3** que existe conectividad exitosa ya que el host 1 con dirección de origen 10.0.0.1 envía un paquete IP

que incluye un *"Echo request"*, y tras la recepción del host 2 con dirección de destino 10.0.0.2, responde con un paquete de datos que contiene un *"Echo reply"* como se aprecia en la **Figura 4-3.**

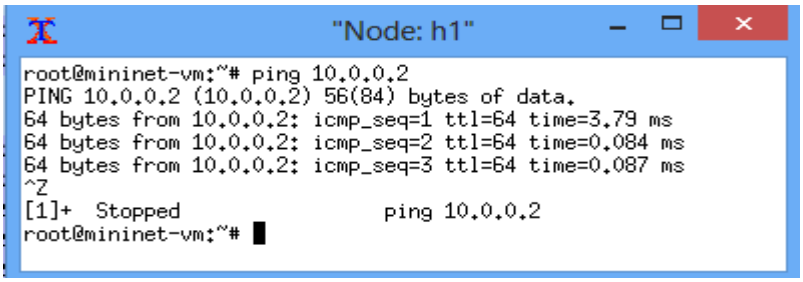


Figura 3-3. Prueba de Ping entre host 1 y host 2
Realizado por: GUERRERO, Daniela. 2017

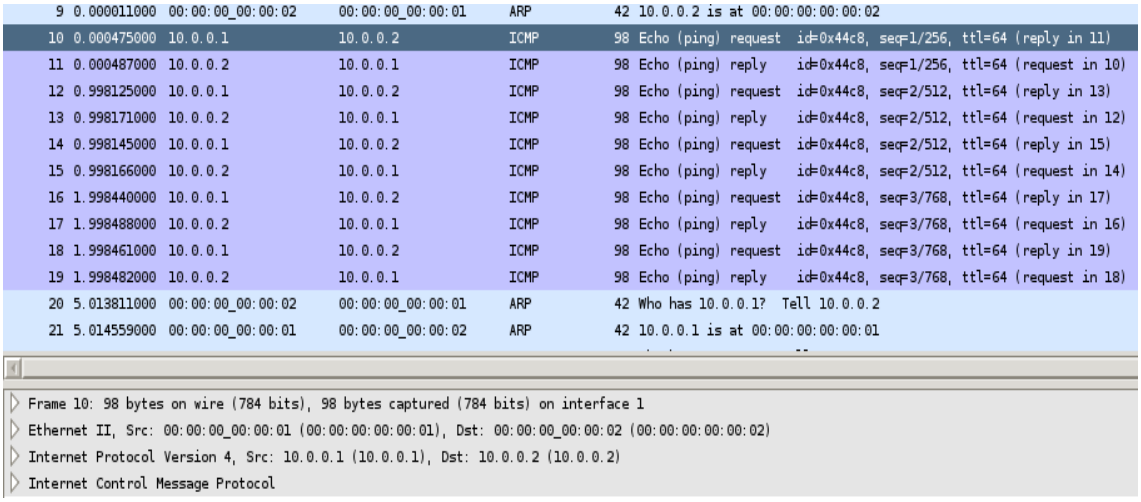


Figura 4-3. Captura Tráfico ICMP (h1 y h2)
Realizado por: GUERRERO, Daniela. 2017

Sucede lo mismo entre los hosts 1, 2 y 3 gracias a las reglas de QoS, en las siguientes figuras se verifica las pruebas de ping.

3.2.1.2 *Prueba de Ping entre host 1 y host 3*

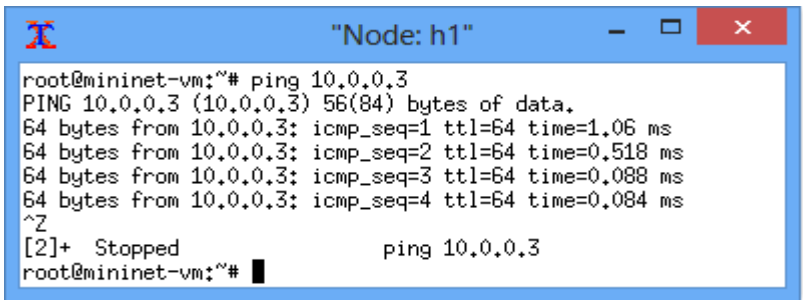


Figura 5-3. Prueba de Ping entre host 1 y host 3
Realizado por: GUERRERO, Daniela. 2017

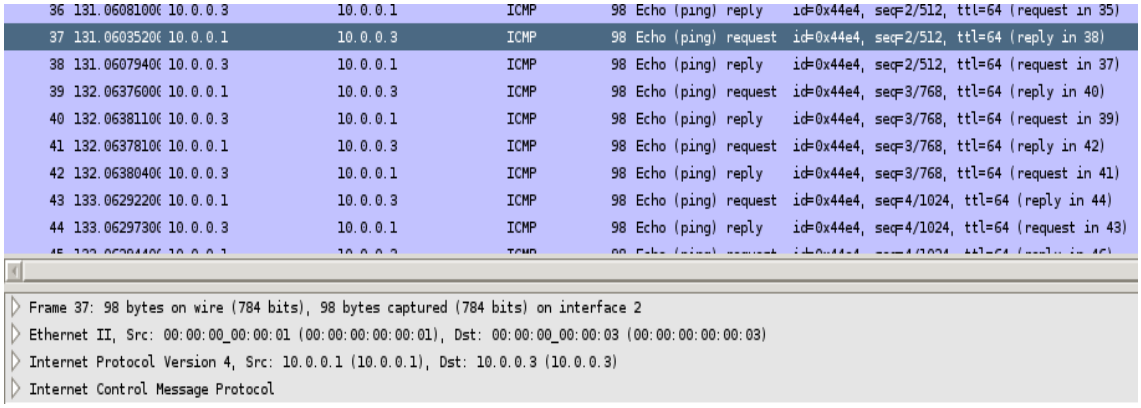


Figura 6-3. Captura Tráfico ICMP (h1 y h3)
Realizado por: GUERRERO, Daniela. 2017

3.2.1.3 *Prueba de Ping entre host 2 y host 1*

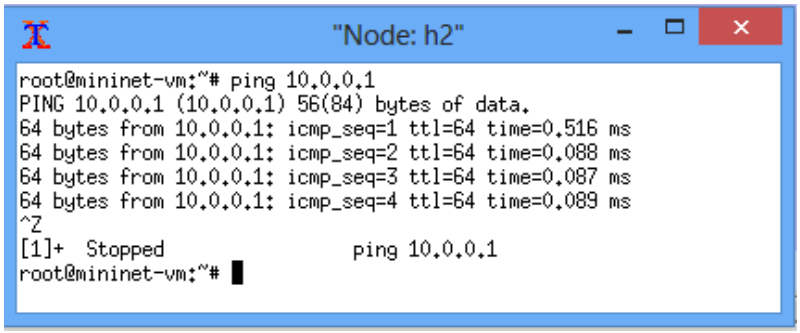


Figura 7-3. Prueba de Ping entre host 2 y host 1
Realizado por: GUERRERO, Daniela. 2017

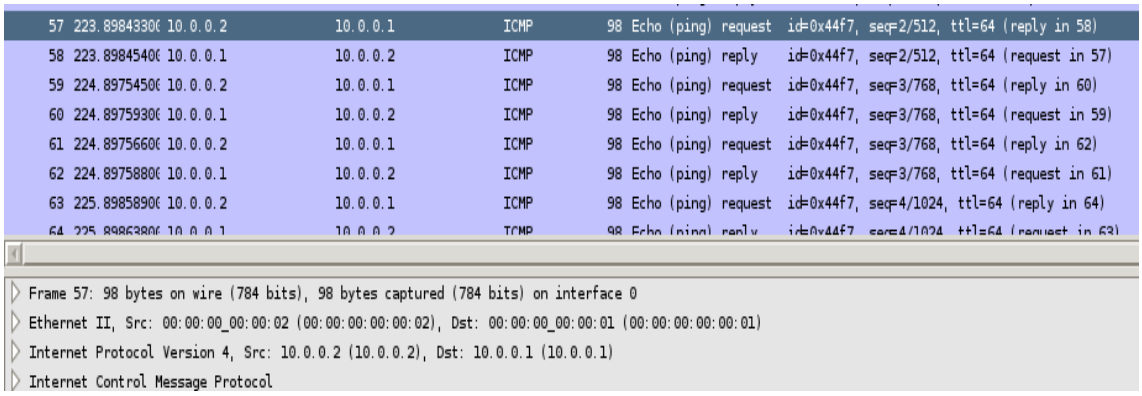


Figura 8-3. Captura Tráfico ICMP (h2 y h1)
Realizado por: GUERRERO, Daniela. 2017

3.2.1.4 *Prueba de Ping entre host 2 y host 3*

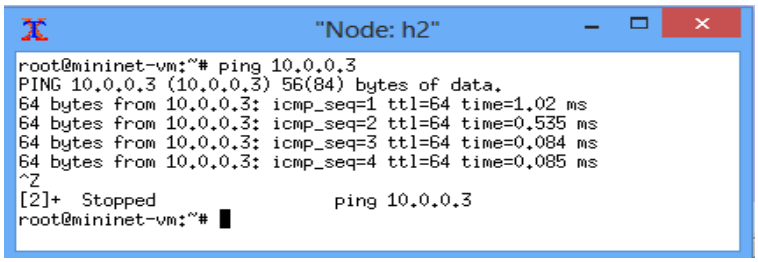


Figura 9-3. Prueba de Ping entre host 2 y host 3
Realizado por: GUERRERO, Daniela. 2017

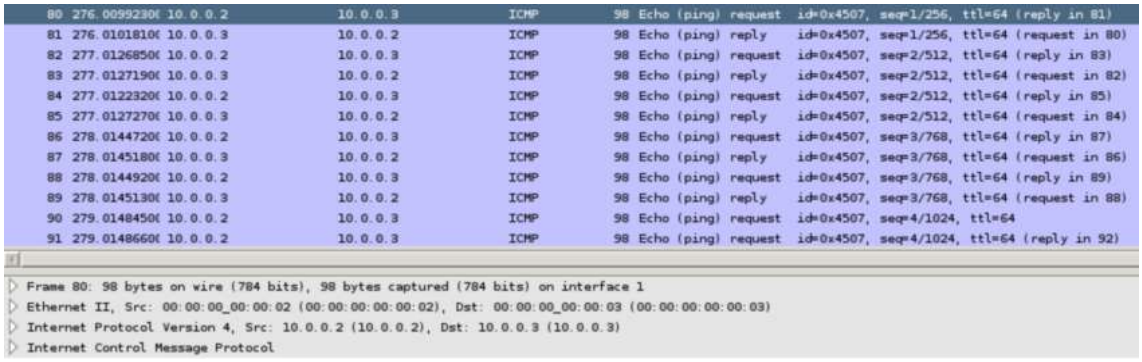


Figura 10-3. Captura Tráfico ICMP (h2 y h3)
Realizado por: GUERRERO, Daniela. 2017

3.2.1.5 *Prueba de Ping entre host 3 y host 2*

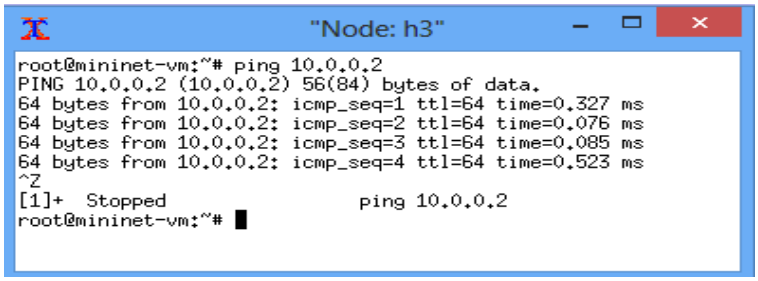


Figura 11-3. Prueba de Ping entre host 3 y host 2
Realizado por: GUERRERO, Daniela. 2017

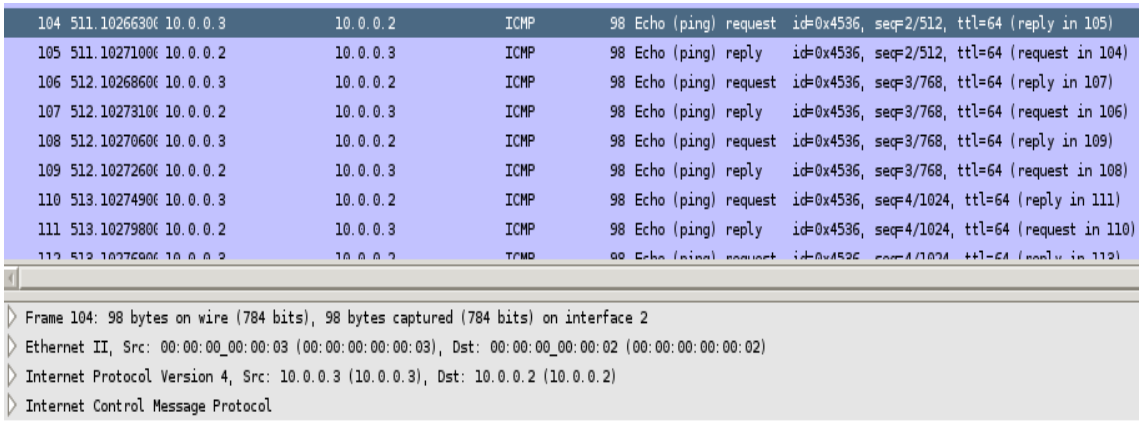


Figura 12-3. Captura Tráfico ICMP (h3 y h2)
Realizado por: GUERRERO, Daniela. 2017

3.2.1.6 *Prueba de Ping entre host 3 y host 1*

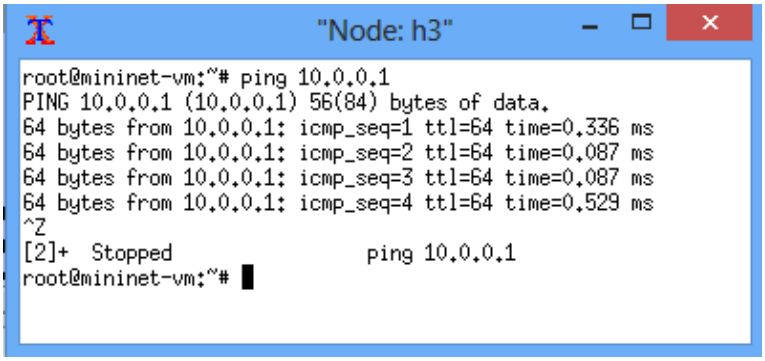


Figura 13-3. Prueba de Ping entre host 3 y host 1
Realizado por: GUERRERO, Daniela. 2017

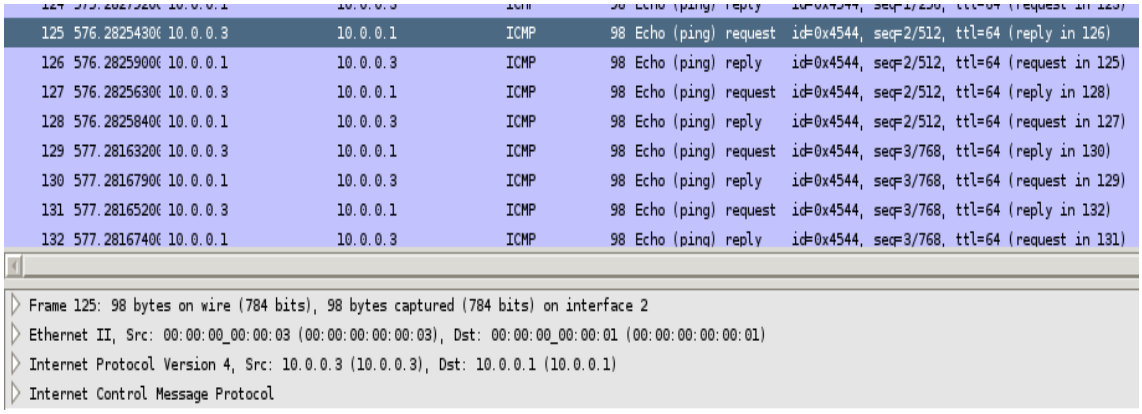


Figura 14-3. Captura Tráfico ICMP (h3 y h1)
Realizado por: GUERRERO, Daniela. 2017

3.2.2 *Pruebas de Ping entre los Hosts: 4 y 5.*

A continuación, se comprueba la conexión fallida de ping entre los hosts 4 y 5, ya que como se puede observar en la **Figura 15-3**, la dirección de origen 10.0.0.4 del host 4 envía en paquete IP incluído *"Echo request"*, sin tener recepción por parte de la dirección de destino 10.0.0.5 del host 5. Se genera en el controlador el resultado de bloqueo de paquetes ICMP como se muestra en la **Figura 16-3.**

3.2.2.1 *Prueba de Ping entre host 4 y host 5*

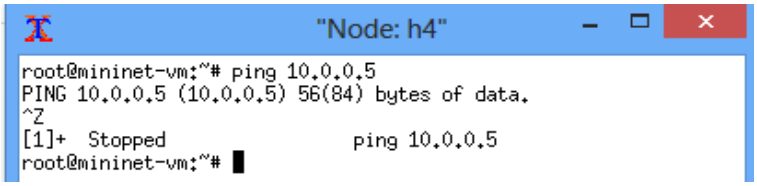


Figura 15-3. Prueba de Ping entre host 4 y host 5
Realizado por: GUERRERO, Daniela. 2017

```
[FW][INFO] dpid=0000000000000001: Blocked packet = ethernet(dst='00:00:00:00:00:
05',ethertype=2048,src='00:00:00:00:00:04'), ipv4(csum=34546,dst='10.0.0.5',flag
s=2,header_length=5,identification=40878,offset=0,option=None,proto=1,src='10.0.
0.4',tos=0,total_length=84,ttl=64,version=4), icmp(code=0,csum=37728,data=echo(d
ata='7\xa0\xf3Y\xffI\n\x00\x08\t\n\x0b\x0c\r\x0e\x0f\x10\x11\x12\x13\x14\x15\x16
\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567',id=17747,seq=5),t
ype=8)
```

Figura 16-3. Bloqueo tráfico ICMP (h4 y h5)
Realizado por: GUERRERO, Daniela. 2017

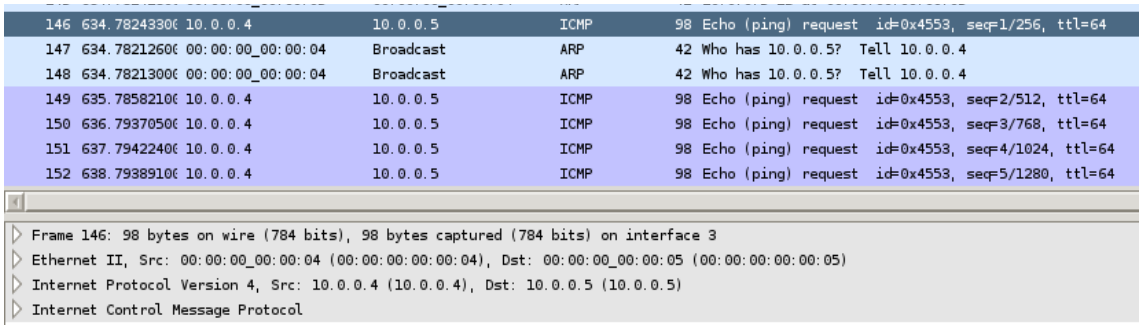


Figura 17-3. Captura Tráfico ICMP (h4 y h5)
Realizado por: GUERRERO, Daniela. 2017

3.2.2.2 *Prueba de Ping entre host 5 y host 4*

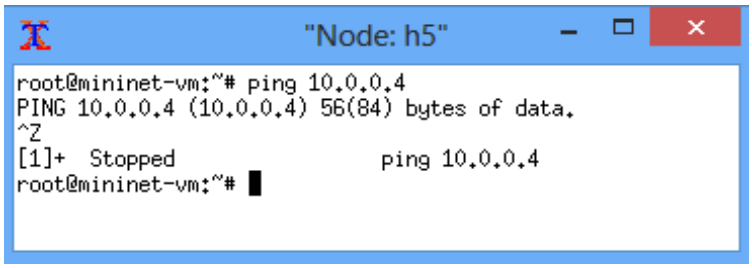


Figura 18-3. Prueba de Ping entre host 5 y host 4
Realizado por: GUERRERO, Daniela. 2017

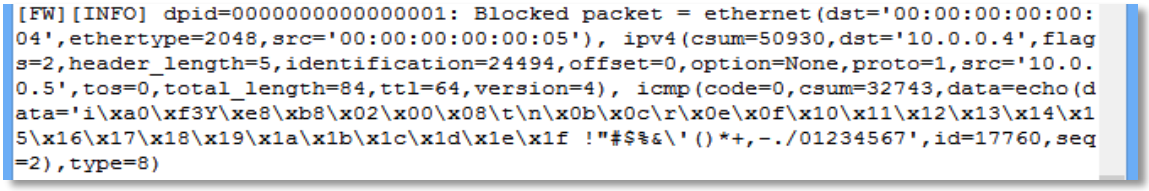


Figura 19-3. Bloqueo tráfico ICMP (h5 y h4)
Realizado por: GUERRERO, Daniela. 2017

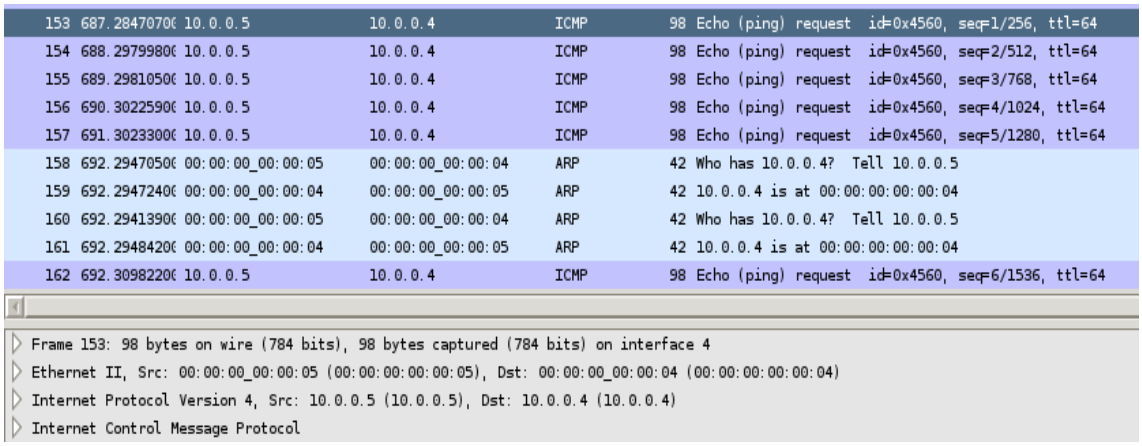


Figura 20-3. Captura Tráfico ICMP (h5 y h4)
Realizado por: GUERRERO, Daniela. 2017

3.2.3 ***Prueba de conectividad con HTTP***

3.2.3.1 *Prueba HTTP entre host 1 y host 2*

Los paquetes que no sean ping y que se transmitan entre los hosts 1, 2 y 3, son bloqueados. Por ejemplo, en la **Figura 21-3** se observa que si se ejecuta *wget* desde la dirección de origen 10.0.0.1 hacia la dirección de destino 10.0.0.2, y en la **Figura 22-3** se observa que se genera en el switch un registro que indica que los paquetes fueron bloqueados.

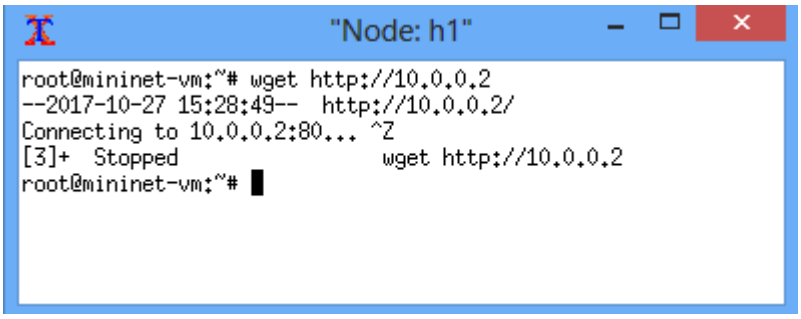


Figura 21-3. Prueba petición http entre host 1 y host 2
Realizado por: GUERRERO, Daniela. 2017

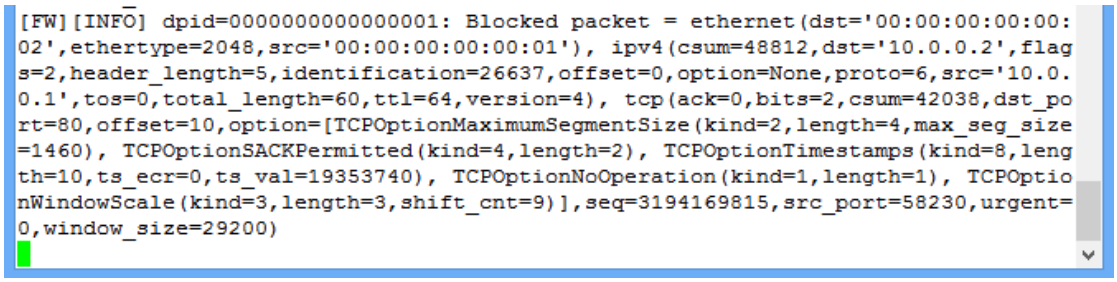


Figura 22-3. Bloqueo petición http (h1 y h2)
Realizado por: GUERRERO, Daniela. 2017

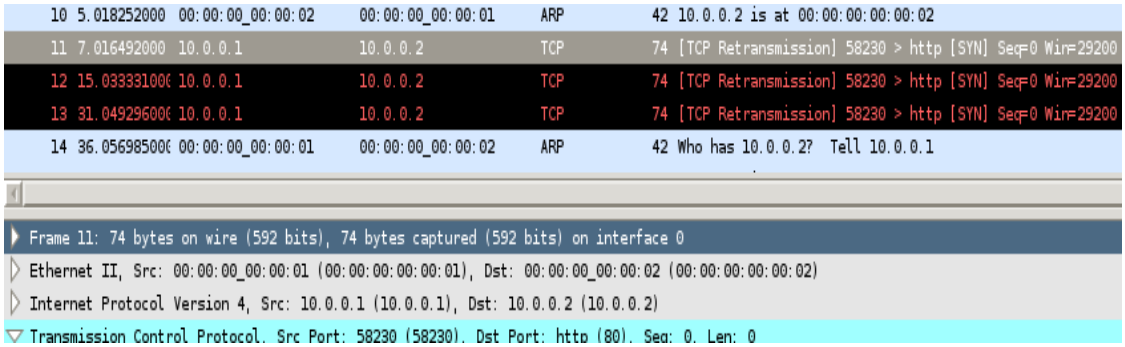


Figura 23-3. Captura Tráfico TCP (h1 y h2)
Realizado por: GUERRERO, Daniela. 2017

3.2.3.2 *Prueba HTTP entre host 4 y host 5*

Entre el host 4 y host 5 se permite el transporte de paquetes que no sean ping como se muestra en la **Figura 24-3,** es por ello que el controlador no emite ningún aviso de bloqueo ya que no está denegando la transmisión de paquetes.

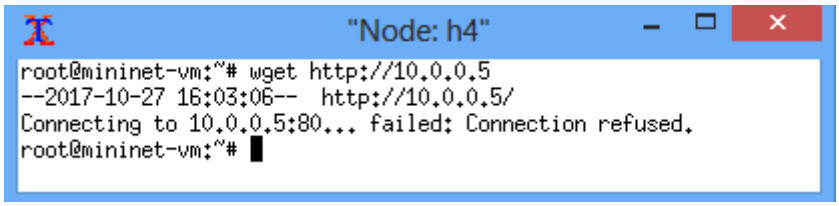


Figura 24-3. Prueba acceso http entre host 4 y host 5
Realizado por: GUERRERO, Daniela. 2017

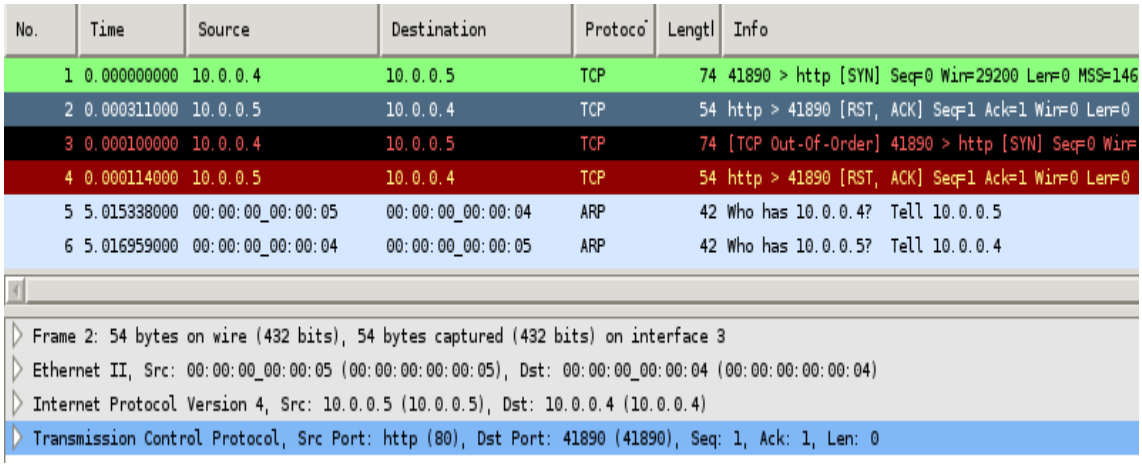


Figura 25-3. Captura de tráfico de acceso http (h4 y h5)
Realizado por: GUERRERO, Daniela. 2017

3.2.4 *Prueba de acceso mediante SSH*

3.2.4.1 *Prueba de conectividad mediante SSH entre host 4 y host 5*

Se permite que el host 4 con nivel de prioridad 10, que transmita paquetes que no sean ping hacia el host 5, por lo que el controlador no emite ningún aviso.

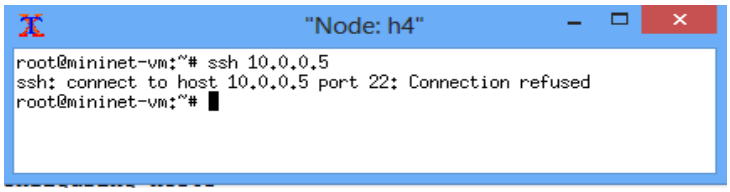


Figura 26-3. Prueba acceso SSH entre host 4 y host 5
Realizado por: GUERRERO, Daniela. 2017

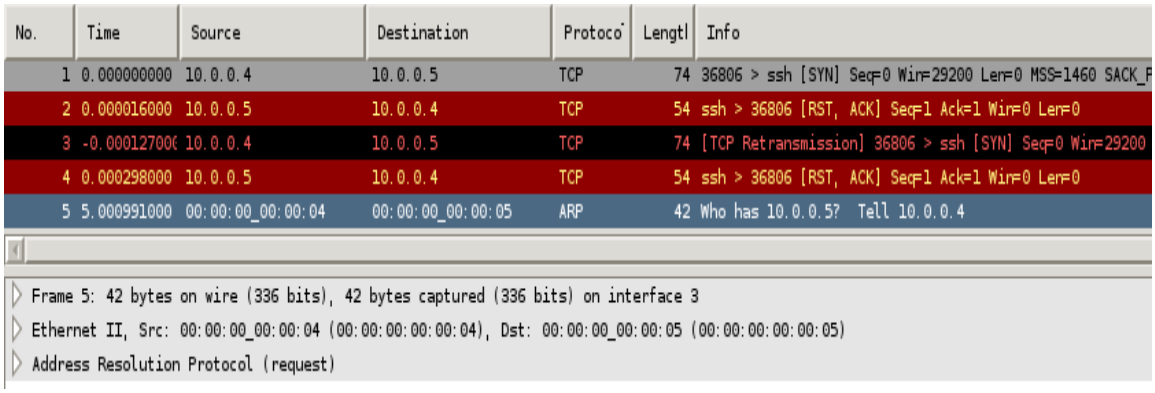


Figura 27-3. Captura tráfico de acceso SSH (h4 y h5)
Realizado por: GUERRERO, Daniela. 2017

3.2.4.2 *Prueba de conectividad mediante SSH entre host 4 y host 1*

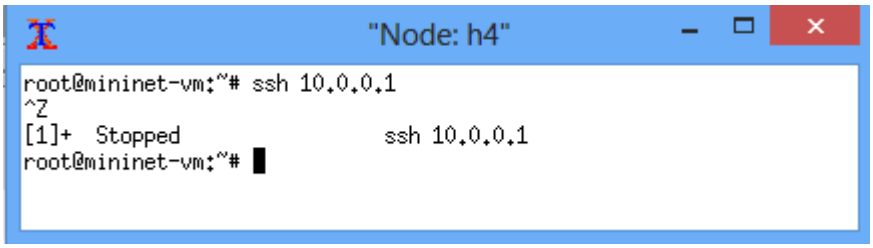


Figura 28-3. Prueba acceso SSH entre host 4 y host 1
Realizado por: GUERRERO, Daniela. 2017

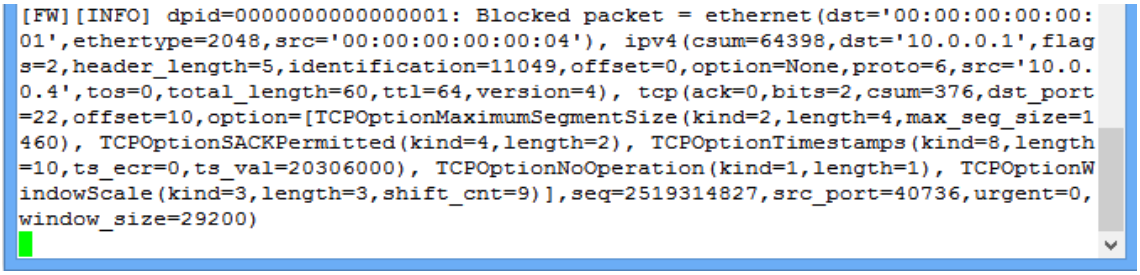


Figura 29-3. Bloqueo acceso SSH (h4 y h1)
Realizado por: GUERRERO, Daniela. 2017

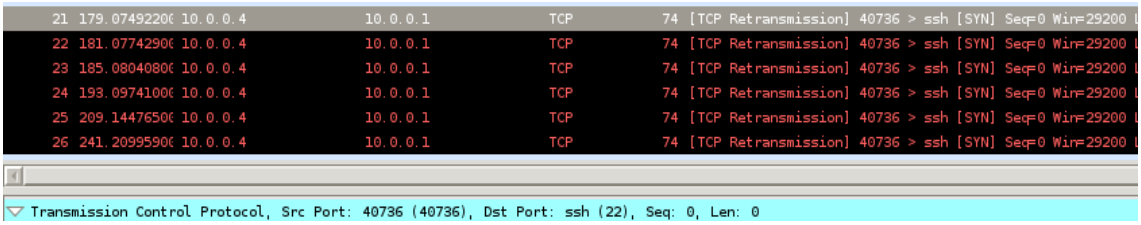


Figura 30-3. Captura tráfico de acceso SSH (h4 y h1)
Realizado por: GUERRERO, Daniela. 2017

CONCLUSIONES

Las Redes Definidas por Software han demostrado a lo largo de este trabajo, que son un nuevo paradigma en las redes de comunicaciones, una tecnología innovadora que pretende ser la solución a todos los problemas de las actuales estructuras de red, es un proyecto ambicioso debido a que se quiere revolucionar la forma como se hace Networking.

En el presente trabajo se proporciona una clara visión de lo que significa la implementación de la red definida por software SDN, ya que gracias a la arquitectura SDN se permitió el mejoramiento de la capacidad de gestión por parte del administrador de red mediante el uso de recursos de estándar abierto, permitiendo así que la red sea escalable y flexible al momento de utilizar en este caso el controlador Ryu.

Se ha seleccionado el controlador Ryu para la red definida por software SDN, ya que se ajusta a las necesidades de virtualización y funcionalidad de la red, el cual permite desarrollar y experimentar con las aplicaciones de Ryu, a pesar de no tener una interfaz gráfica y la documentación sea un poco escasa, hay que recalcar que el desarrollador debe tener una idea básica del lenguaje de programación Python.

Mininet, es muy ágil en cuanto a velocidad de creación y puesta en marcha de topologías SDN, ya que a pesar de no poseer de un entorno gráfico su implementación tarda pocos segundos en realizarse y las pruebas se desarrollan de manera ágil y rápida mostrando potencial al ejecutar cualquier tipo de comando o aplicación.

En el diseño de red para este trabajo, se ha introducido reglas que permiten y deniegan tráfico ICMP, se cambia de prioridad a los flujos entre hosts y en la etapa de pruebas se verificó el correcto funcionamiento del cambio de propiedades de los paquetes en el 100% de las pruebas realizadas en QoS sobre Ryu, posteriormente se realiza la captura de tráfico mediante Wireshark verificando el comportamiento de los flujos en la red implementada.

En la red implementada se dio prioridad y se permitió cierto tipo de tráfico en hosts específicos, para generar y emular un ambiente en que se haga un uso adecuado de los recursos de datos, esto conllevó al aumento del rendimiento y escalabilidad de la red, gracias a las reglas de comportamiento que se introdujeron en el controlador evitando la vulnerabilidad en flujos de datos en hosts con mayor prioridad.

RECOMENDACIONES

Se recomienda tomar en cuenta que los prerrequisitos para el desarrollo de este trabajo deben ser los conocimientos básicos sobre el sistema Operativo Linux para así poder alcanzar un buen control de la arquitectura de Red, además también se debe analizar y socializar las políticas en las que se verán inmerso los usuarios con sus distintos dispositivos de la red.

El emulador Mininet a pesar de su fácil de utilización, se recomienda estudiar la documentación para el entendimiento de sus funcionalidades ya que esta herramienta presenta tutoriales introductorios que permiten desarrollar habilidades técnicas necesarias para el uso del mismo.

Se recomienda que para emular una red en Mininet, se necesita correr un controlador que a su vez ejecute una aplicación, si se ejecuta una topología hay que asegurarse de la utilización de un software switch, el mismo que utiliza una versión específica de OpenFlow, por lo cual debe comprobarse que la versión sea soportada tanto por el controlador, la aplicación y el software switch.

Se recomienda que para mejorar la visibilidad del comportamiento de los hosts se pueda utilizar la función xterm y en cada ventana del host creado se pueda comprobar funcionalidades como: conectividad, transferencia de datos y funcionalidad de la red configurada, así como también el control del tráfico mediante Wireshark que también utiliza una ventana xterm.

BIBLIOGRAFÍA

ALVAREZ, Raúl. *Estudio de las Redes Definidas por Software mediante el desarrollo de escenarios virtuales basados en el Controlador OpenDaylight.* [En línea] [**Tesis] [Maestría**]**.** Universidad Politécnica de Madrid, Escuela Técnica Superior de Ingenieros de Telecomunicación. Madrid - España. (2015). p. 12
[Citado el: 05 de Julio del 2017.]
Recuperado de: http://oa.upm.es/42968

BUILD SDN AGILELY *Ryu SDN Framework* [En línea] 2017
[Citado el: 24 de Septiembre del 2017.]
Disponible en: https://osrg.github.io/ryu/

CHICO, Juan. *Implementación de un prototipo de una red definida por software (SDN) empleando una solución basada en hardware.* [En línea] [**Tesis Pregrado**]**.** Escuela Politécnica Nacional. Facultad de Ingeniería en Sistemas. Quito-Ecuador. (2013). pp. 1-2
[Citado el: 05 de Julio del 2017.]
Recuperado de:

El futuro de las redes inteligentes, SDN Architecture [En línea] 2014
[Citado el: 10 de Agosto del 2017.]
Disponible en: http://www.ramonmillan.com/tutoriales/sdnredesinteligentes.php

Enabling innovation in campus network [En línea] 2008
[Citado el: 4 de Septiembre del 2017.]
Disponible en: http://archive.openflow.org/documents/openflow-wp-latest.pdf

Fundación Open Networking [En línea] 2017
[Citado el: 04 de Agosto del 2017.]
Disponible en: https://www.opennetworking.org/sdn-definition/

HENAO, José. Guía de implementación y uso del emulador de redes MININET. [En línea] [**Tesis Pregrado**] Universidad Tecnológica de Pereira. Facultad de Ingenierías. Pereira-Colombia. 2015b, pg. 11
[Citado el: 26 de Septiembre del 2017.]

Recuperado de:
http://repositorio.utp.edu.co/dspace/bitstream/handle/11059/5713/00678H493%20_Anexo.pdf;jsessionid=BB6EA556B6AEB32CA0AF6F32F625154D?sequence=2

IBÁÑEZ, Federico & LÉVANO, Alejandro & NIETO, Erick. *Diseño e implementación de una herramienta de visualización para análisis en tiempo real de redes SDN/OPENFLOW.* [En línea] [**Tesis Pregrado**]. Universidad Complutense de Madrid. Facultad de Informática. Madrid-España. (2015). p. 14
[Citado el: 5 de Agosto del 2017.]
Recuperado de: http://eprints.ucm.es/38709/

ISA, Rubén. *Implementación de un laboratorio virtual para aprendizaje de redes SDN.* [En línea] [**Tesis Pregrado**]. Universidad de Cantabria. Escuela Técnica Superior de Ingenieros Industriales y de Telecomunicación. Cantabria-España. (2016). pp. 5-6
[Citado el: 24 de Septiembre del 2017.]
Recuperado de:
https://repositorio.unican.es/xmlui/bitstream/handle/10902/9377/387880.pdf?sequence=1

LLAULET, Adriá. *Configuración de un entorno de emulación que permita el diseño, desarrollo y evaluación de Software-Defined Networks con Calidad de Servicio.* [En línea] [**Tesis Pregrado**]. Universidad Politécnica de Catalunya. Facultad de Informática de Barcelona. Barcelona-España. 2016. pp. 32-33

[Citado el: 26 de Septiembre del 2017.]

Recuperado de:
https://upcommons.upc.edu/bitstream/handle/2117/81460/104342.pdf?sequence=1&isAllowed=y

Marca el futuro del networking. [En línea] 2013
[Citado el: 25 de Julio del 2017.]
Disponible en: http://www.networkworld.es/sdn/sdn-marca-el-futuro-del-networking

MARTINEZ, Jackson. *Estudio del funcionamiento de la herramienta Mininet.* [En línea] [**Tesis Pregrado**] Universidad Católica de Pereira. Facultad de Ciencias Básicas e Ingenieria. Pereira-Colombia. 2015, pg. 34
[Citado el: 27 de Septiembre del 2017.]

Recuperado de:
http://repositorio.ucp.edu.co:8080/jspui/bitstream/10785/3665/1/CDMIST115.pdf

MORILLO, Diana. *Implementación de un prototipo de una red definida por software (SDN) empleando una solución basada en Software.* [En línea] [**Tesis Pregrado**]**.** Escuela Politécnica Nacional. Facultad de Ingeniería Eléctrica y Electrónica. Quito-Ecuador. (2014). pp. 140-142
[Citado el: 05 de Julio del 2017.]
Recuperado de:

PUENTE, Jesús. *Algoritmo de calidad de Experiencia para transmisiones de video en Redes Definidas por Software.* [En línea] [**Tesis Pregrado**]**.** Universidad Complutense de Madrid. Madrid-España. (2015)a. p.47
[Citado el: 16 de Septiembre del 2017.]
Recuperado de: http://eprints.ucm.es/27110/

Ryubook [En línea] 2017
[Citado el: 27 de Septiembre del 2017.]
Disponible en: https://osrg.github.io/ryu/resources.html#books

The OpenFlow [archivo PDF] 2013
[Citado el: 10 de Septiembre del 2017.]
Disponible en: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf

TINAJERO, Edwin. *Implementación de un prototipo de switch OpenFlow de bajo costo utilizando una raspberry pi.* [En línea] [**Tesis Pregrado**]**.** Escuela Politécnica Nacional. Facultad de Ingeniería Eléctrica y Electrónica. Quito-Ecuador. (2016), pg. 22-23.
[Citado el: 25 de Septiembre del 2017.]
Recuperado de: http://bibdigital.epn.edu.ec/bitstream/15000/16737/1/CD-7333.pdf

UBUNTU - [en línea] 2017
[Citado el: 5 de Julio del 2017.]
Disponible en: https://www.ubuntu.com/server

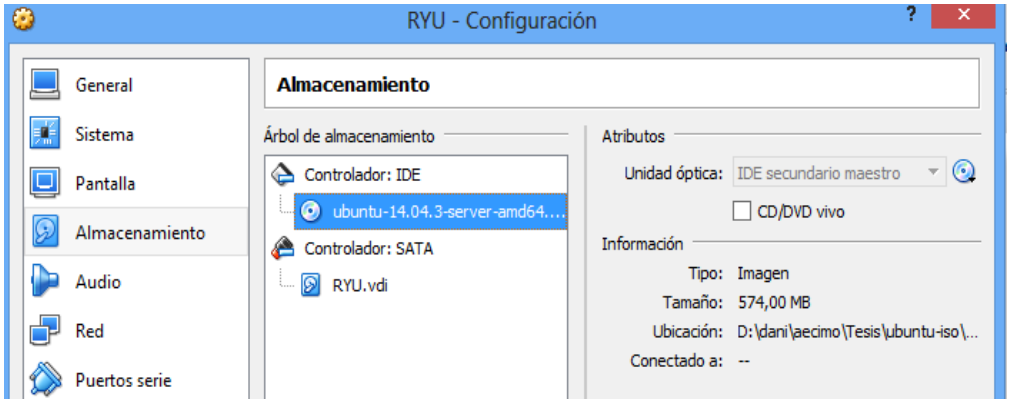
YANEZ, Catherine & GALLLEGOS, Fabián. *Implementación de un prototipo de Red Definida por Software para el HOTSPOT-ESPOCH mediante un controlador basado en*

OpenFlow. [En línea] [**Tesis Pregrado**]**.** Escuela Superior Politécnica de Chimborazo. Facultad de Ingeniería en Sistemas. Riobamba-Ecuador. (2015). p. 107
[Citado el: 05 de Julio del 2017.]
Recuperado de: http://dspace.espoch.edu.ec/handle/123456789/4388

ANEXO A

INSTALACIÓN DEL SO UBUNTU

Imagen ISO: ubuntu-14.04.3-server-amd64
URL: www.ubuntu.org



Se necesita instalar por defecto el servidor SSH para poder trabajar más adelante con el software Putty.

ANEXO B

INSTALACIÓN CONTROLADOR RYU

Instalación Ryu desde Github

```
#sudo apt-get install git
#git clone git://github.com/osrg/ryu.git
```

Cuando el repositorio esté listo se realiza los últimos pasos a continuación:

```
# cd ryu
# sudo python ./setup.py install
# sudo pip install six --upgrade
# cd /usr/local/lib/python2.7/dist-packages/ryu/app
```

ANEXO C

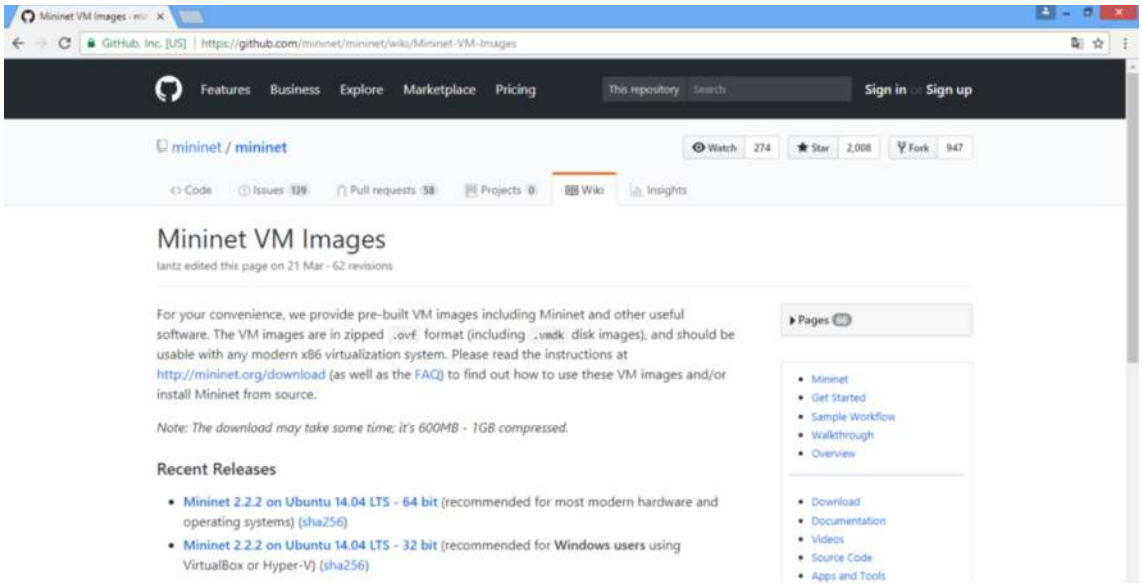
INSTALACIÓN SERVIDOR XMING

Versión Xming: 6.9.0.38
URL: https://xming-x-server.uptodown.com

ANEXO D

EMULADOR MININET

URL: https://github.com/mininet/mininet/wiki/Mininet-VM-Images



ANEXO E

INSTALACIÓN WIRESHARK

```
sudo apt-get install wireshark
```