

UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE CUENCA

CARRERA DE INGENIERÍA DE SISTEMAS

Trabajo de titulación previo a
la obtención del título de
Ingeniero de Sistemas

PROYECTO TÉCNICO:

“Despliegue de una red SDN aplicando el protocolo MPLS y generando políticas de QoS para servicios de telefonía IP.”

AUTORES:

Mesías Mauricio Fernández Mora

Roberth Fernando Ulloa Banegas

TUTOR:

Ing. Pablo Leonidas Gallegos Segovia

Cuenca – Ecuador

SEPTIEMBRE, 2016

DECLARATORIA DE RESPONSABILIDAD

Cuenca, septiembre del 2016

Nosotros, Mesías Mauricio Fernández Mora con C.I. N° 0105109631 y Roberth Fernando Ulloa Banegas con C.I. N° 0105183115 del “DESPLIEGUE DE UNA RED SDN APLICANDO EL PROTOCOLO MPLS Y GENERANDO POLÍTICAS DE QOS PARA SERVICIOS DE TELEFONÍA IP” certificamos que el total contenido de este Proyecto Técnico son de nuestra exclusiva responsabilidad y autoría, en donde hemos consultado las referencias bibliográficas incluidas en el presente documento.



.....
Mesías Mauricio Fernández Mora

0105109631



.....
Roberth Fernando Ulloa Banegas

0105183115

CESIÓN DE DERECHOS DE AUTOR

Cuenca, septiembre del 2016

Nosotros Mesías Mauricio Fernández Mora, con documento de identificación N° 0105109631, y Roberth Fernando Ulloa Banegas, con documento de identificación N° 0105183115 manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del trabajo de grado intitulado: “DESPLIEGUE DE UNA RED SDN APLICANDO EL PROTOCOLO MPLS Y GENERANDO POLÍTICAS DE QOS PARA SERVICIOS DE TELEFONÍA IP.”, mismo que ha sido desarrollado para optar por el título de Ingeniero de Sistemas, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en nuestra condición de autores nos reservamos los derechos morales de la obra antes citada. En concordancia, suscribimos este documento en el momento que hacemos entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Politécnica Salesiana.

.....
Mesías Mauricio Fernández Mora

0105109631

.....
Roberth Fernando Ulloa Banegas

0105183115

CERTIFICACIÓN

Cuenca, septiembre del 2016

Yo declaro que bajo mi tutoría, fue desarrollado el trabajo de titulación: “DESPLIEGUE DE UNA RED SDN APLICANDO EL PROTOCOLO MPLS Y GENERANDO POLÍTICAS DE QOS PARA SERVICIOS DE TELEFONÍA IP”, realizado por: Mesías Mauricio Fernández Mora y Roberth Fernando Ulloa Banegas, obteniendo este Proyecto Técnico que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana para ser considerado como Trabajo de Titulación



.....
Pablo Leonidas Gallegos Segovia

0102593589

AGRADECIMIENTO

Agradezco primero a Dios por la vida y salud para culminar esta meta, a mis padres y hermanos por el apoyo incondicional que me han dado durante todo este trayecto de mis estudios, a mi tutor del proyecto Ing. Pablo por la paciencia y dedicación que nos ha dado durante el seguimiento del mismo.

Mi agradecimiento para mi compañero de trabajo de titulación Roberth, por la ayuda y dedicación durante el proyecto, así también a mis compañeros de clase que me han apoyado.

Mesías Fernández

AGRADECIMIENTO

Agradezco primeramente a Dios por la salud y la guía para terminar una meta tan importante, también agradezco a mis padres que fueron fuente de ánimos y esfuerzo, a mis hermanos por la comprensión y el apoyo incondicional, a mi compañero de trabajo de titulación por la dedicación y la ayuda durante todo este proceso, a mi director de trabajo por haber estado pendiente de este proyecto en todo momento y la atención brindada para culminar con el mismo.

Robertth Ulloa

DEDICATORIA

Dedico de manera especial a mis padres, abuelos y hermanos quienes son el pilar importante en mi vida, gracias a sus consejos, comprensión que siempre me han dado.

Este nuevo logro es gracias a ustedes, he logrado terminar una meta más que al inicio me parecía imposible.

Mesías Fernández

DEDICATORIA

Dedico este trabajo de titulación a Dios, a mis padres y hermanos. A Dios porque me ha guiado durante toda mi vida cuidándome y brindándome la fortaleza para continuar, a mis padres quienes a lo largo de mi vida han velado por mi bienestar incondicionalmente y me han inculcado la importancia del estudio siendo mi apoyo incondicional sin ninguna duda en todo instante convirtiéndose en mi pilar fundamental, sin ellos no hubiese podido conseguir lo que hasta ahora. Finalmente a mis hermanos con quienes he compartido varias experiencias y momentos gratos, enseñando y aprendiendo varios valores importantes para la formación personal como ser humano.

Robertth Ulloa

Contenido

RESUMEN	14
Abstract.....	15
INTRODUCCIÓN	16
OpenFlow.....	16
SDN.....	16
MPLS	17
VoIP	19
Calidad de servicio.....	20
Zabbix	21
OpenStack.....	21
Arquitectura.....	21
Red como Servicio (NaaS).....	22
Red Híbrida.....	22
Mininet.....	22
PROBLEMA	24
ANTECEDENTES	24
IMPORTANCIA Y ALCANCES.....	24
DELIMITACIÓN	25
OBJETIVOS	26
OBJETIVO GENERAL.....	26
OBJETIVOS ESPECÍFICOS.....	26
MARCOTEÓRICO	27
Redes Definidas por software (Software Define Network SDN)	27
OpenFlow.....	27
Calidad de servicio.....	28
MPLS	29
MARCO METODOLÓGICO	31
METODOLOGIA.....	31
Topología	31
Instalación de OpenDayLight (ODL)	32
Pasos para ejecutar OpenDayLight.....	32

1.- Instalar Java JDK	32
2.- Descargar OpenDayLight	32
3.- Ejecutar ODL.....	32
4. Instalación de las características en ODL	33
Ejecución del simulador Mininet	35
1.- Descargar Mininet.....	35
2.- Ejecutar Mininet.....	35
3.- Verificar la topología creada con el comando <i>dump</i>	36
4.- Verificar en el controlador que se creó la topología	36
Configuración de los dispositivos Mikrotik routerboard rb2011u.....	36
1.- Crear Bridge.....	36
2.- Asignar puertos al Bridge	37
3.- Asignar Dirección IP al Bridge.....	37
4.- Creación de flujos (Flows).....	38
5.- Asignar Puertos al Flujo creado.....	38
Instalación de OpenStack.....	39
1.- Actualización de repositorios:.....	39
2.- Instalar git	39
3.- Clonación de OpenStack.....	40
4.- Configurar parámetros para integración de Neutrón con OpenStack	40
5.- Iniciar Instalación.....	40
6.- Verificación de la instalación.....	40
7.- Ingresar al ambiente web	41
Configuración de OpenStack	42
1.- Creación del router neutrón	42
2.- Creación de la red local.....	43
3.- Creación de una Subred	44
4.- Creación de un pool DHCP.....	45
5.- Subir imagen de Ubuntu	45
6.- Creación de llaves para las instancias	47
7.- Creación de políticas de seguridad	48
8.- Creación de la instancia	49

9.- Verificación de la red creada	51
10.- Creación de rutas.....	52
Integrar OpenStack - Neutrón con el controlador OpenDayLight.....	53
1.- Reiniciar el servicio openvswitch	53
2.- Configuración del puente.....	53
3.- Crear conexión para el controlador ODL.....	53
Configuración de flujos para administrar el tráfico con ODL y conmutadores.....	55
Configuración de MPLS	58
RESULTADOS	60
Rendimiento en el servidor VoIP.....	60
Rendimiento de OpenStack.....	61
Tráfico de datos en el servidor Voip.....	62
Tráfico de datos en la nube OpenStack.....	63
Pruebas de MPLS en SDN	64
Prueba de calidad de servicio.....	65
Prioridad de flujos en el controlador.....	65
Flujos instalados en el dispositivo Mikrotik sdn1 (tabla OpenFlow)	65
Flujos instalados en el dispositivo Mikrotik sdn2 (tabla OpenFlow)	66
Comparaciones de QoS.....	66
Trafico de flujos con menor prioridad (5).....	66
Trafico de flujos con mayor prioridad (15).....	67
CONCLUSIONES	68
RECOMENDACIONES	68
Bibliografía.....	69
ANEXOS	71

ÍNDICE DE FIGURAS

Imagen 1 Arquitectura de SDN	17
Imagen 2 Topología básica de la red MPLS.....	18
Imagen 3 Arquitectura del protocolo SIP	20
Imagen 4 Topología de red implementada	31
Imagen 5 Consola principal del controlador ODL	33
Imagen 6 Interfaz de logueo del controlador ODL.....	34
Imagen 7 Interfaz principal del controlador ODL	34
Imagen 8 Creación de una topología en Mininet.....	35
Imagen 9 Verificación de topología creada en Mininet	36
Imagen 10 Topología de Mininet visualizada en ODL	36
Imagen 11 Creación de bridge en Mikrotik.....	37
Imagen 12 Asignación de puertos al bridge	37
Imagen 13 Asignación de IP al bridge en Mikrotik.....	38
Imagen 14 Creación de flows en Mikrotik	38
Imagen 15 Asignación de puertos al flow	39
Imagen 16 IP del servidor y Credenciales para logueo en OpenStack.....	40
Imagen 17 Interfaz de logueo de OpenStack.....	41
Imagen 18 Menú de OpenStack.....	42
Imagen 19 Creación de router Neutrón	42
Imagen 20 Parámetros de creación de router.....	43
Imagen 21 Listado de routers en OpenStack	43
Imagen 22 Opción de creación de redes en OpenStack.....	43
Imagen 23 Parámetros para la creación de una red en OpenStack.....	44
Imagen 24 Parámetros para la creación de una red en OpenStack.....	44
Imagen 25 Parámetros para la creación de una red en OpenStack.....	45
Imagen 26 Listado de redes existentes en OpenStack	45
Imagen 27 Creación de imágenes en OpenStack.....	46
Imagen 28 Parámetros de creación de imágenes en OpenStack.....	46
Imagen 29 Listado de imágenes existentes en OpenStack	47
Imagen 30 Creación de un par de llaves en OpenStack	47
Imagen 31 Listado de llaves existentes en OpenStack.....	47
Imagen 32 Opción de creación de políticas de seguridad para instancias de OpenStack	48
Imagen 33 Parámetros de creación de políticas de seguridad	48
Imagen 34 Listado de políticas de seguridad creadas.....	49
Imagen 35 Opción para la creación de un instancia a partir de una imagen	49
Imagen 36 Parámetros para la creación de una instancia	50
Imagen 37 Parámetros para la creación de una instancia	50
Imagen 38 Parámetros para la creación de una instancia	51
Imagen 39 Listado de las instancias creadas en OpenStack.....	51
Imagen 40 Topología de las instancias y Neutron en OpenStack	52
Imagen 41 Verificación de la interfaz br-ex perteneciente a Neutrón.....	52
Imagen 42 Rutas creadas en Ubuntu para acceso a OpenStack	52

Imagen 43 Estado de conexión de Neutrón con ODL	54
Imagen 44 Topología general del proyecto en ODL	54
Imagen 45 Listado de Dispositivos de la red.....	55
Imagen 46 Módulos de configuración de ODL	55
Imagen 47 Métodos de envío de configuración de ODL.....	56
Imagen 48 Interfaz de configuración de ODL.....	56
Imagen 49 Configuración de instrucciones de ODL	57
Imagen 50 Configuración de prioridad.....	57
Imagen 51 Configuración de Push-MPLS.....	58
Imagen 52 Configuración de pop-MPLS.....	59
Imagen 53 Configuración de etiqueta MPLS	59
Imagen 54 Topología de pruebas.....	60
Imagen 55 Rendimiento del servidor de VoIP	61
Imagen 56 Rendimiento del CPU de OpenStack.....	62
Imagen 57 Tráfico de la interfaz de red en el servidor de VoIP	63
Imagen 58 Tráfico de la interfaz de red en OpenStack	63
Imagen 59 Verificación de MPLS en wireshark	64
Imagen 60 Verificación de MPLS en Wireshark.....	64
Imagen 61 Trafico Openflow en Mikrotik	64
Imagen 62 Configuración de la prioridad.....	65
Imagen 63 Flujos instalados en el router 1	65
Imagen 64 Flujos instalados en el router 2.....	66
Imagen 65 Verificación de prioridades en los flujos instalados.....	66
Imagen 66 Verificación de prioridades en los flujos instalados.....	67

RESUMEN

En la actualidad el creciente número de dispositivos conectados al internet, la demanda masiva de información, los nuevos requerimientos de calidad de servicio, alta disponibilidad y ancho de banda están llevado al colapso de las redes tradicionales.

Los nuevos requisitos de la redes necesitan de una solución viable y eficiente en su diseño y arquitectura mejorando la velocidad de respuesta ante incidentes, ingeniería de tráfico, balanceo de carga, dando paso a un nuevo paradigma denominado redes definidas por software (SDN).

Nuestro proyecto plantea un laboratorio de pruebas, que implementa una arquitectura SDN, en la que consta de un controlador OpenDayLight (ODL) en el que se centraliza la administración y gestión de los dispositivos de reenvió de paquetes, calidad de servicio y el protocolo MPLS. Además añade el protocolo OpenFlow como una característica en conmutadores y enrutadores que homogenizan la red y se integran a SDN.

Además se despliega el servicio de voz sobre IP en una nube privada basada en OpenStack, en el que se aplicaran las pruebas de calidad de servicio. Un conjunto de nodos físicos en dispositivos Mikrotik que soportan openflow nativo complementan nuestra arquitectura.

Abstract

At present the growing number of devices connected to the Internet, the massive demand for information, the new requirements for quality of service, high availability and bandwidth. They are led to the collapse of traditional networks.

The new requirements of the networks need a viable and efficient solution in its design and architecture improving the speed of response to incidents, traffic engineering, load balancing, and giving way to a new paradigm called software defined network (SDN).

Our project plans a testing laboratory, which implements an SDN architecture, which consists of a OpenDayLight controller (ODL) in which the administration and management of devices packet forwarding, Quality of Service and MPLS protocol is centralized. It also adds the OpenFlow protocol as a feature in switches and routers which homogenize the network and integrate SDN.

Besides the Voice over IP in a private cloud based on OpenStack, which tests apply quality of service it is deployed. A set of physical nodes in Mikrotik devices that support native OpenFlow complement our architecture.

INTRODUCCIÓN

OpenFlow

OpenFlow es un protocolo abierto, que permite enviar directamente los flujos de datos en los dispositivos como conmutador o routers tanto físicos y virtuales, por lo que se puede programar la red independientemente de los dispositivos, hoy en día es el protocolo más importante en SDN, al inicio en su versión 1.0 permite solamente la conmutación de paquetes, con el avance en las diferentes versiones se han ido adicionando nuevas funcionalidades, por ejemplo en la versión OpenFlow 1.2 tiene soporte para el protocolo MPLS. [1]

Además en la parte principal de un conmutador controlado por OpenFlow es su tabla de flujo, esta tabla mantiene las reglas que son instalados o actualizados mediante un controlador SDN a través del plano de control intercambiando mensajes OpenFlow establecidos mediante sesiones TCP.

Ejemplo. Cuando un host A envía paquetes a un host B, el conmutador comprueba la entrada en su tabla de flujo, determinando cual es el puerto correcto por donde tiene que reenviar los paquetes para alcanzar al host B.

SDN

Las redes definidas por software (SDN) son redes programables que permiten centralizar la administración de los dispositivos tales como conmutadores enrutadores, puertas de enlaces, separando el plano de control y el plano de datos. En plano de control se encuentra el dispositivo denominado controlador en el que se centraliza la gestión y administración de la red y los flujos de datos que el plano de reenvío infraestructura (Hardware) es el responsable únicamente del reenvío de paquetes. [1]

La arquitectura de SDN se basa en tres capas, que son:

- Capa de aplicación: Consiste en las aplicaciones de negocio que serán usadas por los clientes o usuarios de la red, por ejemplo VoIP.
- Capa de control: A esta capa pertenece el controlador SDN como OpenDayLight (ODL) el mismo que será el responsable del plano de control de la red y se comunicará con los nodos a través del protocolo OpenFlow.
- Capa de infraestructura. Está formada por los dispositivos de la red como los enrutadores o conmutadores que son los que reenvían los paquetes para alcanzar el destino.

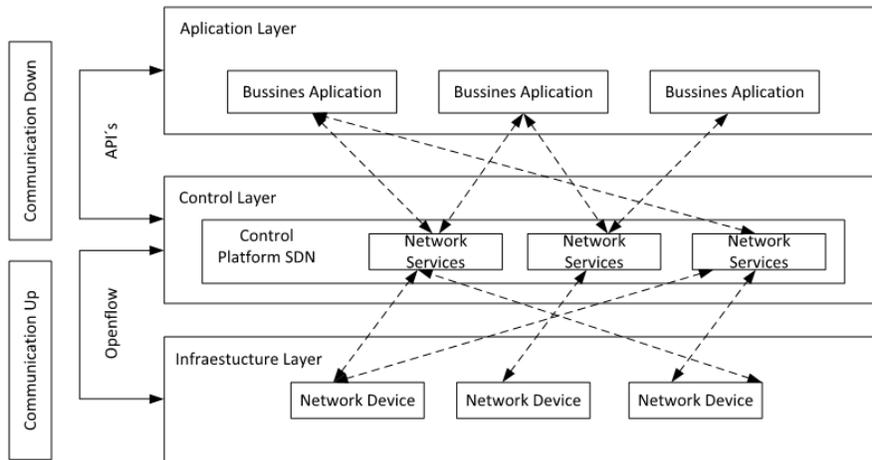


Imagen 1 Arquitectura de SDN

La interface hacia el norte (nortbound) permite comunicarse al controlador SDN con una gran variedad de aplicaciones de negocio que forman parte de la red, mediante el uso de interfaces API (Interfaces de Programación de Aplicaciones).

La interface hacia el sur (southbound) está especificada por el protocolo OpenFlow permite la comunicación entre el controlador SDN y los enrutadores o conmutadores de la red, permitiendo hacer cambios dinámicos de las configuraciones de acuerdo a los requerimientos de los usuarios.

De esta manera SDN hace uso del protocolo OpenFlow como estándar de las redes definidas por software.

SDN cambia el entorno de las redes tradicionales retirando la “inteligencia” de los dispositivos activos como enrutadores y conmutadores, dejándolos en el plano de reenvío de paquetes. Centraliza las funciones “inteligentes de la red” como enrutamiento, control de ancho de banda, protocolos, alta disponibilidad e ingeniería de tráfico en el controlador.

De esta manera reduce el tiempo de respuesta a incidentes, el costo de administración, vuelve más ágil y eficiente el despliegue de soluciones de TI, integrándose a múltiples plataformas de hardware a través de un API como es el caso de Cisco, Juniper, Mikrotik, Dell, etc.

MPLS

MPLS (Multiprotocol Label Switching) Este protocolo trabaja entre la capa de enlace de datos y la capa de red del modelo OSI. Se le conoce también como un protocolo de capa intermedia o capa 2.5 del modelo OSI.

El funcionamiento básico de MPLS consiste en agregar una etiqueta en los paquetes que ingresen a la red para que de esta manera los dispositivos, al leer el valor de dicha etiqueta sepan a donde tienen que direccionar los datos sin analizar por completo la cabecera haciendo más rápido el transporte sin importar el tipo de tráfico que ingrese a los dispositivos.

Una etiqueta es un identificador único que se le asigna a los paquetes de datos para poder enrutarlo dentro de la red MPLS.

Además separa la manera en la que se transportan los paquetes de los servicios ofrecidos en la red, para esto es necesario realizar una codificación de instrucciones en las cabeceras de los paquetes.

MPLS mantiene circuitos virtuales así como también un estado de la comunicación entre dos nodos.

Los nodos MPLS constan de dos planos, el plano de reenvío de MPLS y el plano de control MPLS, estos nodos pueden realizar enrutamiento en capa 3 y conmutación en capa 2 además de la conmutación de paquetes etiquetados. [2]

Para que una red MPLS funcione correctamente debe contar con tres elementos que hacen posible que las etiquetas sean enviadas a través de dicha red:

- LSR (Label Switching Router): Este elemento se encarga de la conmutación de las etiquetas.
- LSP (Label Switched Path): Es el camino por el que los paquetes etiquetados seguirán dentro de la red MPLS.
- LDP (Protocolo de distribución de etiquetas): Distribuyen las etiquetas MPLS entre los equipos de la red.
- LER (Label Edge Router): Dispositivo que se encuentra al inicio y al final de la red MPLS, se encarga de agregar y quitar las etiquetas de los paquetes. [2] [3]

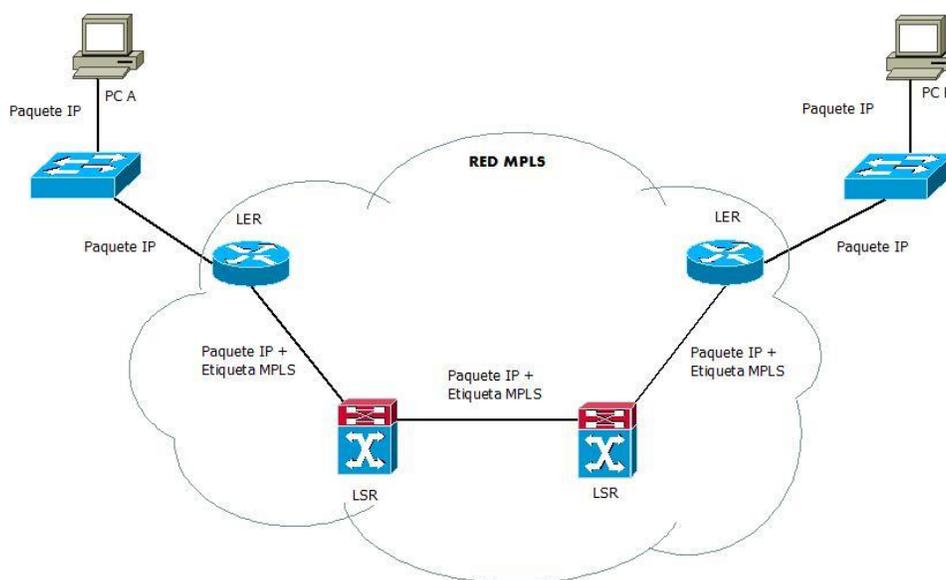


Imagen 2 Topología básica de la red MPLS

VoIP

VoIP (Voz sobre el Protocolo de Internet) está compuesto por varias tecnologías tales como protocolos, dispositivos, codecs, cuya integración permiten la transmisión de voz a través de una red de datos digital.

Un códec es un conjunto de algoritmos los cuales convierten la señal de audio analógica en datos digitales para ser transmitidos por una red IP en forma de paquetes, también cumplen con la función de comprimir la información y de esta manera reducir el ancho de banda durante la transmisión. Algunos de los codecs utilizados para audio son los siguientes: G711 u-a, G.726-32, G.729. [4]

Existen una gran cantidad de protocolos con uso para VoIP, sin embargo los más utilizados por su menor consumo de recursos de red son SIP (protocolo de inicio de sesión) y RTP (Protocolo en Tiempo Real).

El protocolo SIP fue desarrollado por la IETF (Internet Engineering Task Force), se lo emplea para el establecimiento de la conexión tanto con el servidor de VoIP así como también con los usuarios de la red.

La arquitectura del protocolo SIP generalmente está compuesto principalmente por tres componentes:

- Proxy SIP: Este componente es el intermediario entre los clientes, y permite el establecimiento de las llamadas.
- Red IP: La red IP puede ser pública y privada y es el medio a través el cual viajan los paquetes SIP mediante el cual se transmite la llamada entre los clientes.
- Clientes: Son los usuarios que utilizan el servicio de VoIP. [5]

RTP es un protocolo definido por el IETF, es utilizado para la transmisión de voz o fase de conversación [4], trabaja sobre la capa de transporte, al momento de establecerse una llamada un flujo de voz viaja encapsulado en paquetes RTP, el mismo a su vez es encapsulado en un datagrama UDP una vez que el datagrama es entregado al destinatario este decodificará los datos y comenzará a reproducir la llamada estableciendo así la cadena de paquetes. [5]

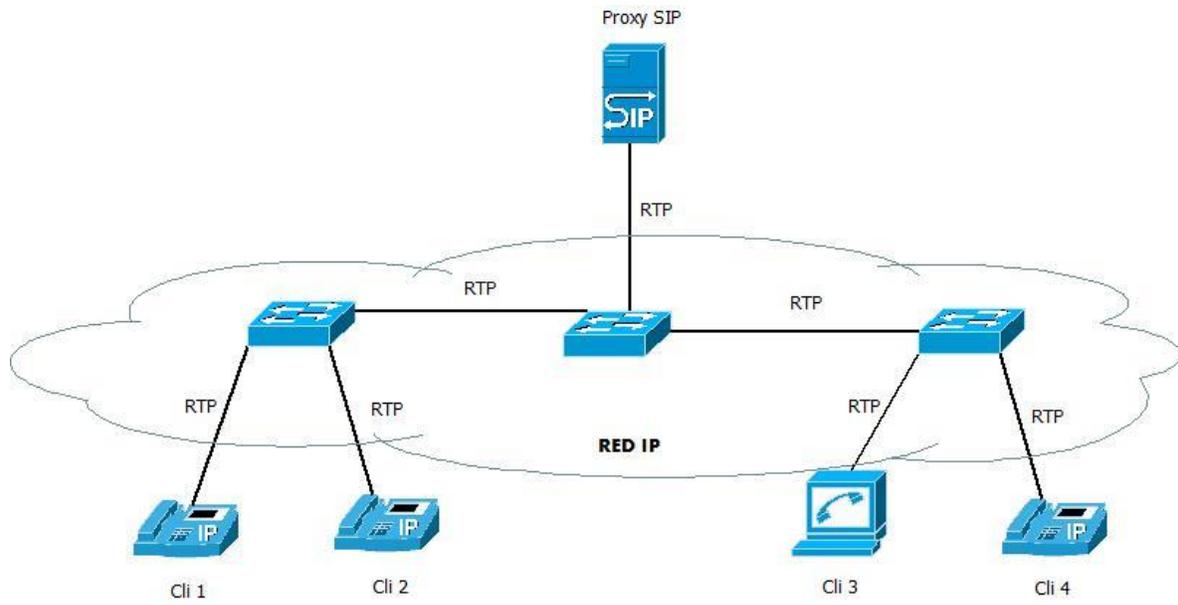


Imagen 3 Arquitectura del protocolo SIP

Calidad de servicio

Las redes de telecomunicaciones transportan diversidad de servicios como datos, voz y video, volviéndose imprescindible para los portadores de servicio así como los proveedores de servicios (ISP) y empresas con infraestructura propia, implementar mecanismos para priorizar el tráfico considerando las políticas y el tipo de datos para asegurar un flujo ágil de los mismos mejorando el rendimiento de la red y satisfaciendo las necesidades del usuario.

La calidad de servicio definida por la IETF (Internet Engineering Task Force), se refiere a un “conjunto de requisitos de servicio que debe cumplir la red durante el transporte de los datos”. [6]

“El objetivo fundamental de cualquier mecanismo de QoS es asegurar que la congestión excesiva en la red no interfiera con los paquetes asegurados con calidad de servicio, ante esto es importante definir que los mecanismos de QoS no crean una capacidad adicional en la red, sino que se prioriza el tráfico y la asignación de capacidad para los paquetes cuando la red se congestiona.”. [7]

Principalmente existen dos maneras de asegurar la calidad de servicios en una red de telecomunicaciones.

Mecanismo de mejor esfuerzo (Best-effort delivery): Se ofrece el mejor servicio posible al usuario en el momento en el cual acceda a la red variando el ancho de banda y el tiempo de respuesta dependiendo de la cantidad de tráfico en la red. [8]

Servicios diferenciados (DiffServ): Este método consiste principalmente en clasificar los paquetes IP de acuerdo a los términos de QoS, es decir que existirán servicios con mayor

prioridad que otros en la red. Como un ejemplo un servicio de voz o video streaming tendrá una mayor preferencia que un servicio de correo. [8]

Zabbix

Zabbix es una herramienta empresarial de software libre diseñada específicamente para el monitoreo de rendimiento y disponibilidad de servidores, máquinas virtuales y dispositivos de la red. [9]

Zabbix ofrece varias opciones de monitorización

- Comprueba la disponibilidad de servicios como HTTP o SNMP (Protocolo Simple de Administración de Red) sin instalar agentes en el equipo monitoreado.
- Maneja un agente zabbix que debe ser instalado en el equipo a monitorear el cual es el encargado de recopilar información del equipo, tales como carga del CPU, memoria utilizada
- Tiene soporte para el protocolo SNMP

OpenStack

OpenStack es una solución de cloud computing de código abierto.

Su objetivo principal es el de proveer una solución flexible para nubes públicas y privadas considerando dos requerimientos básicos:

- Las nubes deben ser simples de implementar. [10]
- Las nubes deben ser masivamente escalables. [10]

Una característica importante de esta solución para la nube es que se ajusta a las necesidades de quien desee implementarlo.

Arquitectura

OpenStack tiene una arquitectura muy grande para poder ofrecer un servicio confiable a la hora de implementarlo, a continuación se presentan sus componentes:

En la figura 2 se pueden visualizar los siguientes componentes:

- Dashboard (Horizon): Provee una interfaz a los usuarios y administradores.
- Compute (Nova): Recupera imágenes y transforma las solicitudes de los usuarios en instancias o máquinas virtuales.

- Network (Neutrón): Provee redes virtuales como servicio entre dispositivos administrados por otros servicios de OpenStack, también permite la creación de redes y posteriormente asociarlas entre instancias de redes privadas así como también el acceso desde redes públicas.
- Block Storage (Cinder): Permite el almacenamiento de las máquinas virtuales alojadas en la nube.
- Image (Glance): Provee un catálogo y repositorio para las imágenes. [10]
- Object Store (Swift): Permite el almacenamiento de objetos, funciona como un contenedor para el almacenamiento de archivos para recuperarlos posteriormente.
- Identity (Keystone): Permite la autenticación y autorización para todos los servicios de la nube. [10]

Red como Servicio (NaaS)

“Las nubes públicas, privadas e híbridas tienen un requisito común: las redes.” [11]

Un elemento común en todas las nubes es la red de la nube siendo este imprescindible para la conexión de los recursos de la nube con los usuarios.

NaaS es un modelo abstracto de un servicio de red ya sea de capa 2 o capa 3 según el modelo OSI, se encarga de definir como los usuarios acceden a los recursos de la nube, y también como estos recursos están conectados en la nube ya sea privada, pública o híbrida. [11]

Red Híbrida

Al hablar de una red híbrida se hace referencia a una red en la que operan juntos los protocolos de redes tradicionales con los de las redes definidas por software (SDN).

“Una red híbrida hace posible la migración de las redes tradicionales a las SDN debido a la convergencia de los protocolos sin importar el tamaño de la red ni el tipo de dispositivos que se estén usando.” [12]

Para que se combinen las tecnologías tradicionales con las SDN es necesario un middleware o puerta de enlace (Gateway) que integra el transporte del tráfico y los servicios y protocolos.

Mininet

Mininet es un simulador de red basado en Linux que crea redes con enrutadores y conmutadores virtuales compatibles con el protocolo OpenFlow para escenarios de redes

definidas por software. Mininet colabora con el apoyo para crear redes experimentales, antes de ponerlas en ambientes reales [13]

El presente trabajo integra los componentes antes detallados en una arquitectura funcional. SDN es el corazón de la arquitectura y sobre la red subyacentes (Mikrotik) se implementa servicios de telefonía IP que está hospedada en una nube privada basada en OpenStack.

PROBLEMA

ANTECEDENTES

En la actualidad se han generado nuevos escenarios y necesidades en las redes de datos, los mismos que requieren de una dinámica de cambios en estructuras de redes complejas que se integren con conceptos como el internet de las cosas, las redes Celulares de nueva generación 4G y 5G, y que se integren con soluciones en la nube.

Las soluciones tradicionales en infraestructura y administración de red son cerradas, y distribuidas, no permiten los cambios dinámicos, y no se alinean con las necesidades del negocio.

Generando un alto retardo en el despliegue y varios puntos de falla en redes metropolitanas o para las Megas ciudades.

La arquitectura horizontal de las redes definidas por software (SDN) fomenta la interoperabilidad y fragmenta el mercado, potenciando la innovación y generando nuevo modelos de negocio basados en las aplicaciones SDN.

El nuevo paradigma de soluciones de red bajo el termino de definidos por software van inicialmente por las redes definidas por software, la seguridad, y las funciones definidas por software este nuevo entorno va de la mano con las tecnologías emergentes y de un modelo de negocio exitoso como son la computación en la nube y las redes de nueva generación.

IMPORTANCIA Y ALCANCES

Debido a que las redes fueron construidas en la década de los 60 en la actualidad ya no están preparadas para satisfacer los servicios requeridos por los usuarios, como por ejemplo cambios en los patrones de uso de la red debido a los usuarios móviles con nuevas aplicaciones, flexibilidad, escalabilidad, despliegue de nuevos servicios; tampoco son fácilmente administrables ni adaptables a situaciones inesperadas a las que están sujetas las redes actuales.

Las redes definidas por software son de gran importancia ya que minimizan la mayoría de problemas en las redes de datos actuales, la migración del plano de control hacia un controlador central, permite que la toma de decisiones para la conmutación de datos ya no lo realice el dispositivo sino el administrador de la red; el despliegue de servicios es más ágil debido a que los cambios son hechos en software y no en el hardware del dispositivo, permitiendo mayor eficiencia en el uso de los recursos de la red.

Dado que la inteligencia de la red está centralizado en el controlador, este maneja flujos de datos independientes a las marcas de dispositivo de red, logrando así tener interoperabilidad entre diferentes marcas.

El alcance de este proyecto, es implementar un prototipo de red definida por software para servicios de VoIP, en el cual se podrá demostrar la flexibilidad de administración de los dispositivos y cambiar el comportamiento de la red de forma dinámica, obteniendo así una red confiable y segura

La infraestructura de red SDN será implementado mediante el controlador OpenDayLight y el protocolo OpenFlow, el servicio de VoIP estará funcionando en la nube privada utilizando el software OpenStack, también se utilizará Neutrón como un middleware, el mismo que permitirá la interacción entre las redes tradicionales, SDN y la nube es decir se implementará la red como un servicio (NaaS).

Para la implementación de MPLS se utilizará el simulador Mininet para crear una topología de red sobre la cual se configurará el protocolo MPLS con el protocolo OpenFlow versión 1.3, finalmente terminamos realizando las pruebas de rendimiento y analizando los resultados.

DELIMITACIÓN

En este trabajo de titulación se busca demostrar en un ambiente de pruebas los beneficios de la Red Definida por Software (SDN) que es el corazón de nuestra investigación, se implementa un nube privada en la que se levanta dos instancias: la primera es IaaS o infraestructura como servicio en el que se desplegara telefonía sobre IP, la segunda instancia es NaaS la red como servicio, que servirá de puerta de enlace entre la red tradicional o heredada, la nube privada y la SDN.

Se aplica una arquitectura física con equipos Mikrotik para los nodos sobre OpenFlow y los protocolos de red tradicionales, y así mismo nodos virtuales sobre Mininet en los que se implementará MPLS. En el controlador de SDN OpenDayLight, se aplicara las políticas de MPLS y QoS, también se definirán los protocolos de pruebas que se detallara más adelante.

Este proyecto esta implementado para el centro de investigación “Cloud Computing, Smart Cities & High Performance Computing” de la Universidad Politécnica Salesiana sede Cuenca.

OBJETIVOS

OBJETIVO GENERAL

Desplegar una red híbrida, es decir combinar una red tradicional con una red definida por software con la finalidad de implementar dentro de la red los protocolos MPLS y generar políticas de calidad de servicio para la aplicación de VoIP en el protocolo SIP.

OBJETIVOS ESPECÍFICOS

- Desplegar un controlador SDN usando el software open source OpenDayLight.
- Generar un escenario de redes híbrida usando el protocolo OpenFlow y dispositivos Mikrotik.
- Configurar los protocolos MPLS y las políticas de calidad de servicio.
- Implementar un sistema de computación en la nube, generar la instancia de Neutrón, dando como el modelo de la red como servicio (NaaS).
- Interconectar las redes heredadas y las redes SDN.
- Obtener la interoperabilidad en el servicio de VoIP.
- Realizar pruebas de rendimiento en este escenario.
- Integrar Neutrón como Middleware entre las redes heredadas y las redes SDN.

MARCOTEÓRICO

Redes Definidas por software (Software Define Network SDN)

En la actualidad las redes de datos están creciendo de tamaño rápidamente [14] por tal motivo se da la necesidad de adaptarse a las demandas de tráfico en la red de datos, con la infraestructura convencional es ineficaz en su administración. Esta razón da cabida a un nuevo paradigma denominado Redes Definidas por Software (SDN) [15] [16]

Las SDN fue introducido en la década de los 1990 por la universidad de Stanford como un protocolo de pruebas y a para investigación, en la actualidad se ha convertido en una alternativa de mejora de las redes apoyada por grandes industrias de las telecomunicaciones como Cisco Juniper HP, etc. [16]

En SDN se separan el plano de control y el plano de datos [17] [18] el plano de datos es el responsable del reenvío de paquetes es decir se encarga de envío y recepción de estos a través de la red, y el plano de control se encarga de la “inteligencia de la red”, selección de protocolo, balanceo de carga, etc. Para esto implementa el controlador, dicho que configura la operación de los equipos de reenvío, aplicando políticas de encaminamiento, VLAN, calidad de servicio, etc. de los conmutadores y encaminadores. Actualiza las tablas de flujo, y aplica reglas y conjuntos de acciones e instrucciones de la red. La comunicación entre el controlador y el los dispositivos de reenvío se realiza mediante el protocolo OpenFlow, que homogeniza la administración la red y complementa la arquitectura SDN.

Un ejemplo de esta arquitectura se esboza en Open-Vswitch que habilita nodos virtuales para SDN. Estos nodos son responsables de reenviar el tráfico de entrada, el mantenimiento de las tablas de flujo y la comunicación con el controlador [15]

“Al dividir el plano de control y el plano de datos las SDN proporciona una lógica centralizada de administración que permite una gestión flexible que se lleva a cabo con un controlador”. [16] [19].

El despliegue de SDN, requiere que los componentes de hardware soporte un API de integración y soporten OpenFlow, como es el caso de Cisco Nexus 3048. Por ejemplo en un ISP, no se puede reemplazar todos los dispositivos, por costos por que se requiere una interconexión entre la red heredada y la SDN, teniendo como resultado una red Híbrida, tomando como middleware a Neutrón o la red como servicio NaaS [19] [20].

OpenFlow

OpenFlow es un protocolo abierto, que representa el estándar en las SDN. Un flujo OpenFlow se compone de algunos campos que contienen las cabeceras de los paquetes Ip

por ejemplo las direcciones MAC, direcciones IP de los destinos, y un conjunto de acciones a realizar. [21]

Un conmutador OpenFlow mantiene una tabla de flujos para almacenar las reglas de reenvío de paquetes, un canal seguro para comunicarse con el controlador y el protocolo OpenFlow en sí mismo. Cuando llega un paquete de datos al conmutador este comprueba los campos de la cabecera de dicho paquete en busca de coincidencias con las reglas de flujo instaladas de forma secuencial, si los campos coinciden se realiza la acción definida en el flujo. [18] [21]

Mientras tanto, el controlador envía los comandos para gestionar los dispositivos OpenFlow y recoger información de todos ellos, es el responsable de proporcionar información completa acerca de la topología de red y su estado, manteniendo dicha topología de dominio SDN en tiempo real. [20]

SDN tiene la capacidad de integración con OpenStack (computación en la nube) debido a que OpenStack dispone de un módulo denominado Neutrón que cuenta con openswitch que se comporta como intermediario y permite el intercambio de paquetes entre las instancias en la nube con la red SDN. [22]

Calidad de servicio

Las redes transportan una gran cantidad de servicios como datos, voz y video, Los proveedores de servicio (ISP) y empresas con infraestructura propia, implementan mecanismos para dar un tratamiento diferenciado el tráfico según lo requerimientos de sus clientes, considerando la criticidad de los servicios y el tipo de información. Para asegurar un flujo ágil de los mismos mejorando el rendimiento de la red y satisfaciendo las necesidades del usuario.

La calidad de servicio definida por la IETF (Internet Engineering Task Force), se refiere a un “conjunto de requisitos de servicio que debe cumplir la red durante el transporte de los datos”. [6]

“El objetivo fundamental de cualquier mecanismo de QoS es asegurar que la congestión excesiva en la red no interfiera con los paquetes asegurados con calidad de servicio, ante esto es importante definir que los mecanismos de QoS no crean una capacidad adicional en la red, sino que se prioriza el tráfico y la asignación de capacidad para los paquetes cuando la red se congestiona.” [7]

Existen varios protocolos que tienen capacidades de calidad de servicio como es el caso de OpenFlow para las redes SDN y NETCONF (Protocolo de Configuración de Red). En el caso de OpenFlow, este provee calidad de servicio a través de un mecanismo de colas. [23] [24]

Al igual que en una intranet, los usuarios esperan que los servicios ofertados en la nube dispongan de alta disponibilidad incluyendo también requisitos de QoS. [24] [25]

Por otro lado telefonía IP es uno de los servicios que mayor calidad de servicio requieren debido que son muy susceptibles a los retardos de la red y problemas con inducción, delay y jitter.

“Una política de calidad de servicio único pueden incluir una gran cantidad de mecanismos de QoS configurados en varios dispositivos de la red.” [26]

El ciclo de vida de QoS se representa con cinco operaciones principales:

- Creación de colas: En la cual se generan las colas en los dispositivos de la red, en el caso de las redes SDN el protocolo OpenFlow es el encargado de la generación de dichas colas.
- Adición del flujo de QoS: Se asigna una prioridad al tráfico de la red.
- Modificación del flujo de QoS: En este proceso se modifica el ancho de banda requerido de las conexiones de red.
- Supresión de flujo de QoS: Se eliminan los flujos de calidad de servicio en los dispositivos de red, el protocolo OpenFlow es el encargado de esta operación en las redes SDN.
- Supresión de colas: Se eliminan las cosas con el protocolo OpenFlow, es importante recalcar que en las dos últimas operaciones para que esto sea posible, se define un tiempo durante el cual funcionaran o estarán activas las colas y los flujos de calidad de servicio. [27]

MPLS

“Multiprotocol Label Switching (MPLS) integra un marco de intercambio de etiqueta con el enrutamiento de la capa de red. La idea básica consiste en la asignación de etiquetas de longitud fija cortos a los paquetes en la entrada a una red MPLS.” [2]

Las características de MPLS hacen que sea posible la implementación de nuevos servicios que no pueden ser soportados por una red sin este protocolo, debido a esto, varias compañías importantes han empezado a utilizar MPLS para ofrecer una mejor calidad de servicio a sus clientes. [28] [29]

Los nodos MPLS constan de dos planos, el plano de reenvío de MPLS y el plano de control MPLS, estos nodos pueden realizar enrutamiento en capa 3 y conmutación en capa 2 además de la conmutación de paquetes etiquetados. [2]

Para que una red MPLS funcione correctamente debe contar con tres elementos que hacen posible que las etiquetas sean enviadas a través de dicha red:

- LSR (Label Switching Router): Este elemento se encarga de la conmutación de las etiquetas.
- LSP (Label Switched Path): Es el camino por el que los paquetes etiquetados seguirán dentro de la red MPLS.

- LDP (Protocolo de distribución de etiquetas): Distribuyen las etiquetas MPLS entre los equipos de la red. [2] [3]
- LER (Label Edge Router): Son dispositivos que se encuentran al inicio y al final de la red, este es el encargado de agregar y quitar las etiquetas a los paquetes que ingresen a la red MPLS.

MARCO METODOLÓGICO

METODOLOGIA

La metodología aplicada es Scrum. En la que se realiza entregas parciales de los avances de un proyecto, hemos usado esta técnica pues nuestro proyecto se dividió en varias fases, iniciando por la definición de la topología de red, servicios y despliegue de la plataforma.

En los siguientes párrafos detallaremos nuestro trabajo.

Topología

La topología de pruebas a implementar es una red estrella, cuyo eje central son las Redes Definidas por Software y se integra con redes tradicionales mediante un middleware (Neutrón). Desplegamos una nube privada con instancias de IaaS para acceder a servicios de VoIP, y las redes como servicio (NaaS) con Neutrón. El controlador de SDN está basado en el software ODL y los equipos de reenvío y de redes tradicionales en el Hardware Mikrotik como se lo visualiza la figura 4.

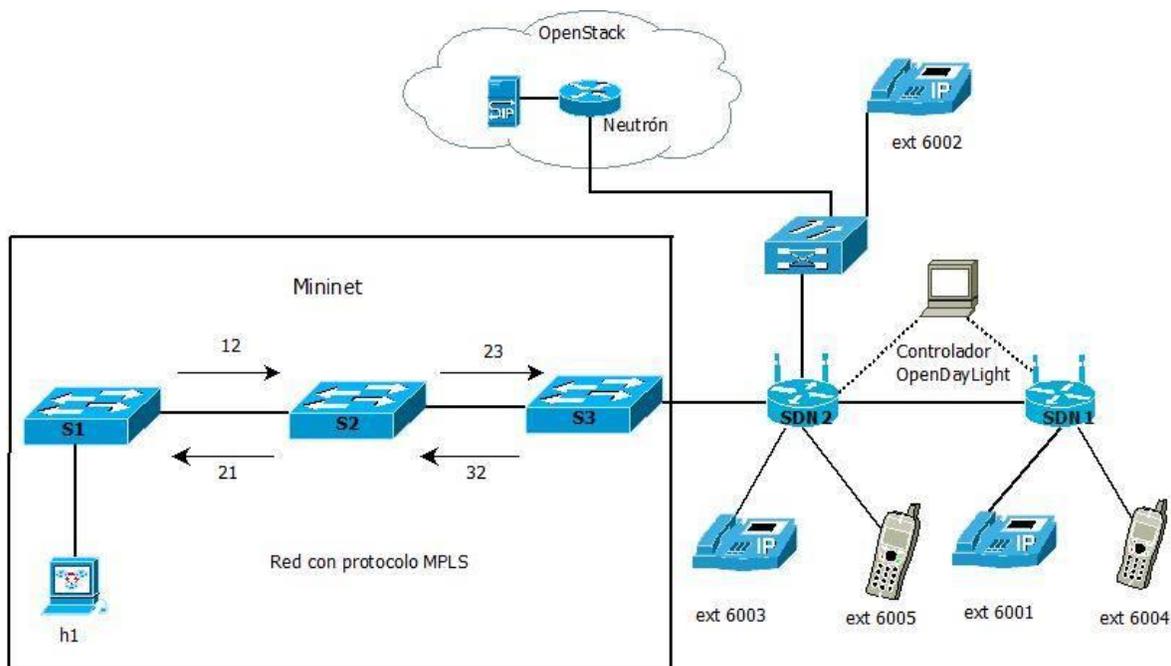


Imagen 4 Topología de red implementada

Instalación de OpenDayLight (ODL)

La instalación de ODL se realizó sobre el sistema operativo Windows 7 de 64 bits en un ambiente físico, las características del equipo utilizado son:

- Procesador Intel CORE I5.
- Memoria RAM de 8 GB.

Pasos para ejecutar OpenDayLight

ODL está escrito en Java y utiliza Maven como herramienta de construcción para ello los requisitos necesarios es tener Java.

1.- Instalar Java JDK.

En este paso es necesario descargar y ejecutar el instalador de JDK de la página oficial de ORACLE.

<http://www.oracle.com/technetwork/es/java/javase/downloads/index.html>

2.- Descargar OpenDayLight

Descargar la distribución de OpenDayLight de la página oficial.

<https://www.opendaylight.org/downloads>

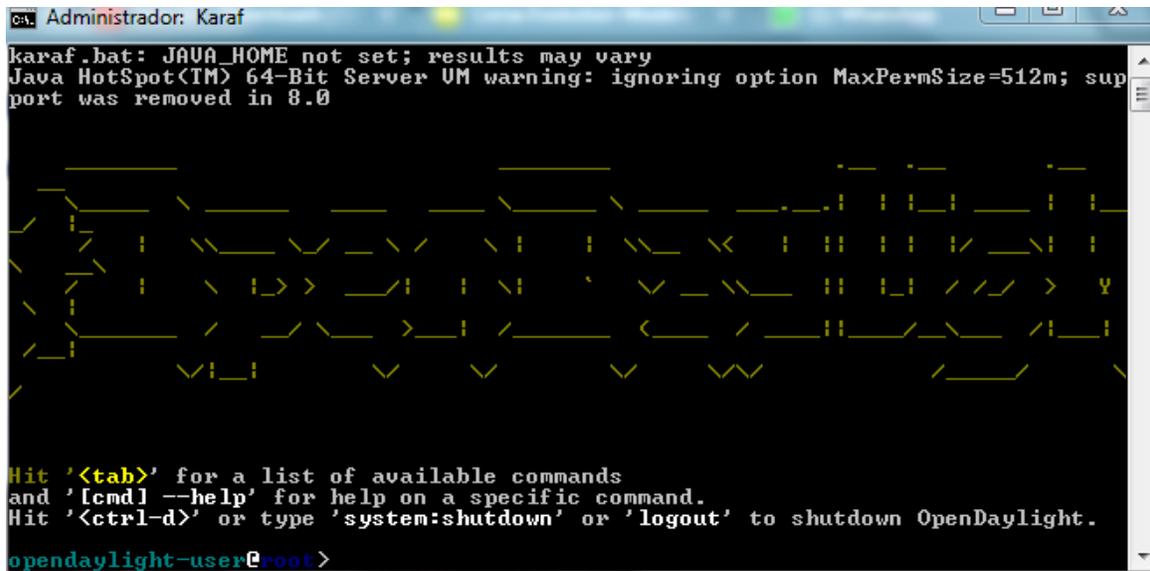
Distribution-karaf-0.4.1-Beryllium-SR1.zip

3.- Ejecutar ODL

Extraer el archivo descargado y ejecutar el archivo KARAF.BAT, este se encuentra dentro de la carpeta:

Distribution-karaf-0.4.1-Beryllium-SR1/bin

Una vez que el controlador se haya ejecutado se mostrará una consola como muestra la imagen 5.

The image shows a terminal window titled "Administrador: Karaf". The output text is as follows:

```
karaf.bat: JAVA_HOME not set; results may vary
Java HotSpot(TM) 64-Bit Server VM warning: ignoring option MaxPermSize=512m; sup
port was removed in 8.0

Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'system:shutdown' or 'logout' to shutdown OpenDaylight.
opendaylight-user@root>
```

Imagen 5 Consola principal del controlador ODL

4. Instalación de las características en ODL

Las características son los diferentes módulos que están disponibles para ODL, están desarrollados para configurar de diferentes maneras la red de datos.

Las características necesarias se las deben instalar de la siguiente manera:

```
feature: install odl-restconf odl-mdsal-apidocs odl-dlux-all odl-dlux-core odl-dlux-node
odl-dlux-yangui odl-l2switch-switch odl-openflowplugin-all
```

Al abrir un navegador e ingresar a la dirección <http://localhost:8181/index.html> se podrá observar una interfaz como se muestra en la imagen 6.

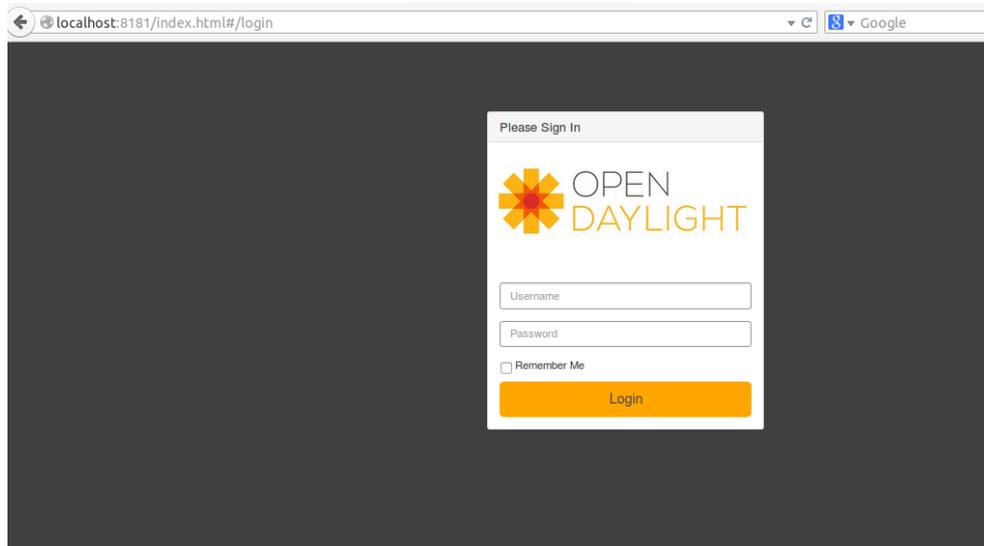


Imagen 6 Interfaz de logeo del controlador ODL

A continuación se debe ingresar las credenciales que OpenDayLight define por defecto
username: admin Password: admin.

Al ingresar con los datos solicitados se podrá observar la interfaz principal de ODL como
se muestra en la imagen 7.

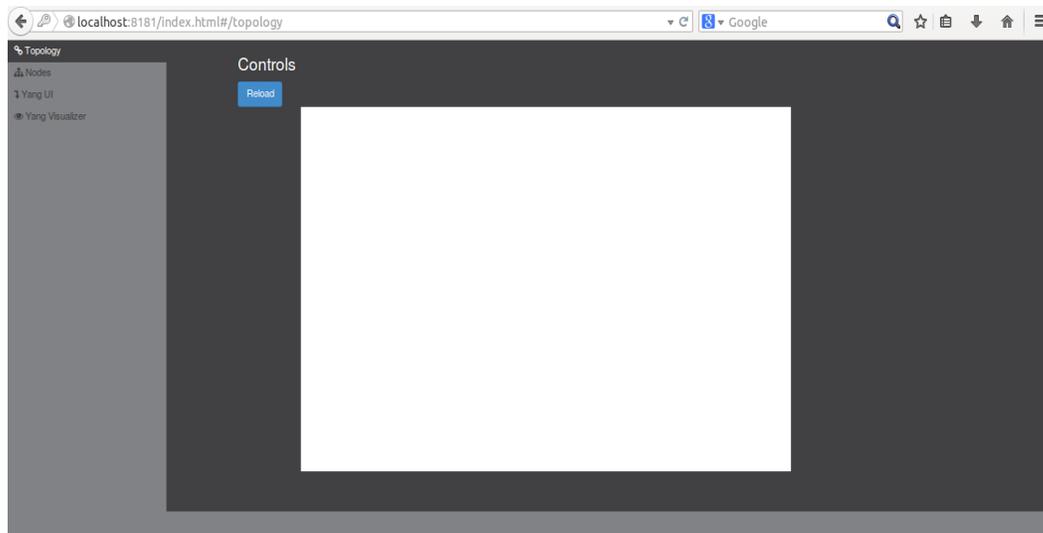


Imagen 7 Interfaz principal del controlador ODL

Ejecución del simulador Mininet

Para la simulación de la red con MPLS es necesario utilizar el simulador denominado Mininet, este software permite crear redes virtuales y configurarla desde el controlador OpenDayLight.

1.- Descargar Mininet

Descargar de la página oficial <https://github.com/mininet/mininet/wiki/Mininet-VM-Images> y ejecutar la imagen descargada en VirtualBox, el sistema operativo de dicha imagen es Ubuntu 14.

Los requerimientos de la máquina virtual para su funcionamiento eficaz es el siguiente:

- 2 GB de memoria RAM.
- Asignar un procesador de 1 núcleo.

2.- Ejecutar Mininet

Para ejecutar Mininet y crear la topología es necesario ingresar el código siguiente en la consola:

```
sudo mn --controller=remote, ip=192.168.10.99,port=6633 --topo=linear,3 --mac --switch ovsk, datapath=user
```

A continuación en la imagen 8 se puede observar el código ingresado para crear la topología, en donde se especifica que el controlador será remoto, la dirección IP del controlador, el puerto de conexión y tipo de topología.

```
mininet@mininet-vm:~$ sudo mn --controller=remote, ip=192.168.10.99, port=6633 --t
opo=linear,3 --link=tc --mac --arp
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (s2, s1) (s3, s2)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet>
```

Imagen 8 Creación de una topología en Mininet

3.- Verificar la topología creada con el comando *dump*

En la imagen 9 se puede observar todos los dispositivos virtuales que se crearon en Mininet anteriormente.

```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=3116>
<Host h2: h2-eth0:10.0.0.2 pid=3119>
<Host h3: h3-eth0:10.0.0.3 pid=3121>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=3126>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None pid=3129>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None pid=3132>
<RemoteController{'ip': '192.168.10.99', 'port': 6633} c0: 192.168.10.99:6633 pid=3110>
mininet>
```

Imagen 9 Verificación de topología creada en Mininet

4.- Verificar en el controlador que se creó la topología

Una vez ejecutado los comandos en Mininet se puede observar la topología creada en el controlador ODL como se muestra en la imagen 10.

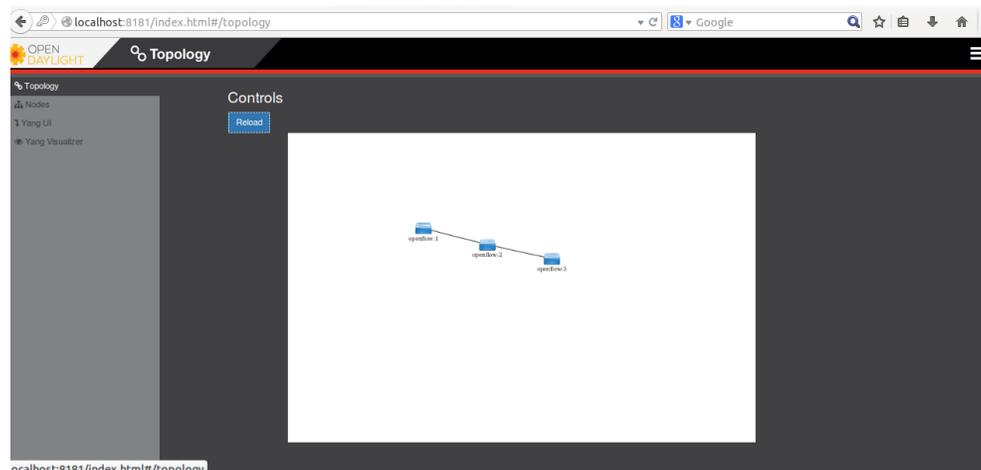


Imagen 10 Topología de Mininet visualizada en ODL

Configuración de los dispositivos Mikrotik routerboard rb2011u

1.- Crear Bridge

En el menú de configuración que se encuentra a la izquierda de la interfaz se debe elegir la opción Bridge, y posteriormente se debe crear el bridge como se observa en la imagen 11.

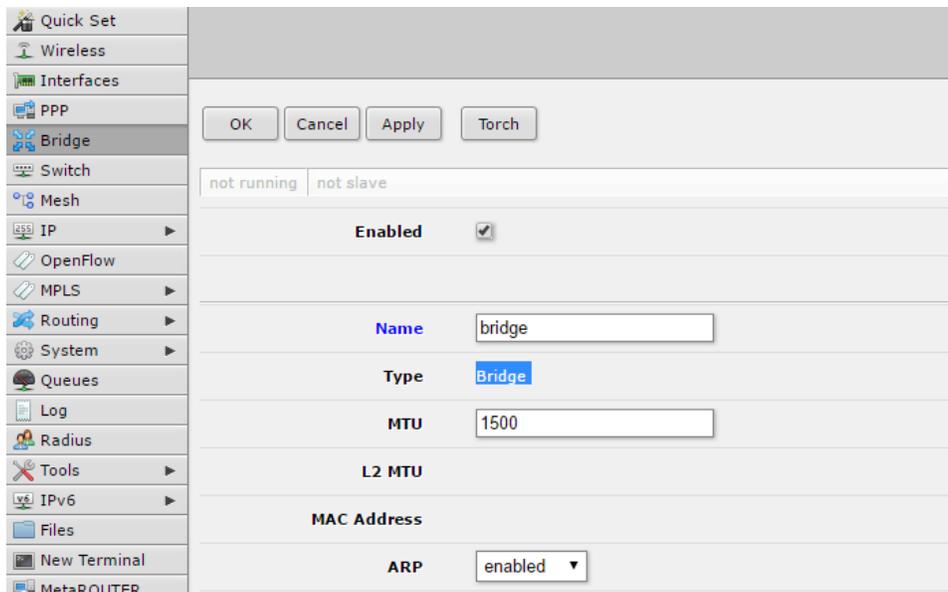


Imagen 11 Creación de bridge en Mikrotik

2.- Asignar puertos al Bridge

Una vez creado el bridge, en la pestaña Ports se deberán seleccionar los puertos que tendrá el bridge, estos puertos conectará el controlador y los otros dispositivos de la red como se indica en la imagen 12.

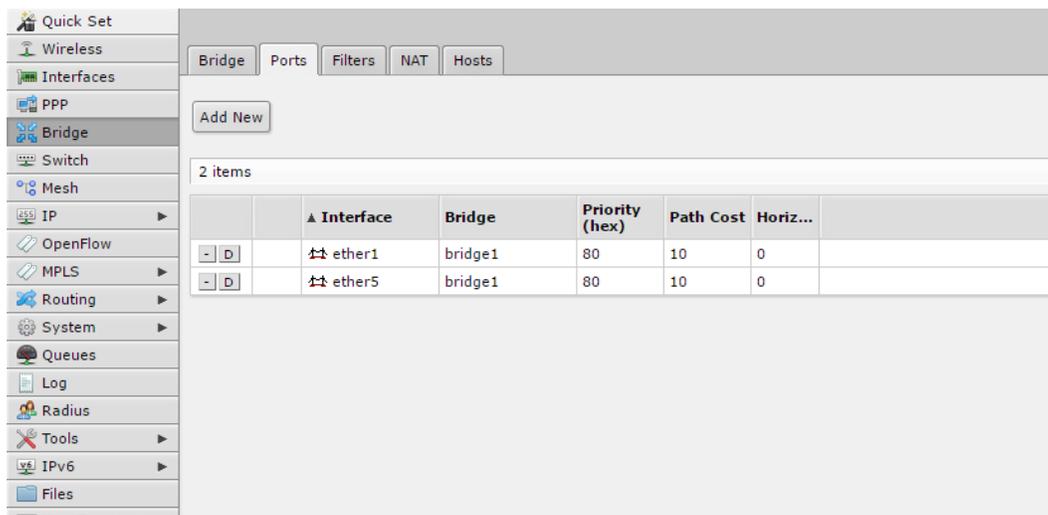


Imagen 12 Asignación de puertos al bridge

3.- Asignar Dirección IP al Bridge

En las opciones del menú izquierdo se debe seleccionar la opción IP => Address y asignar una dirección IP al bridge de acuerdo a la imagen 13.

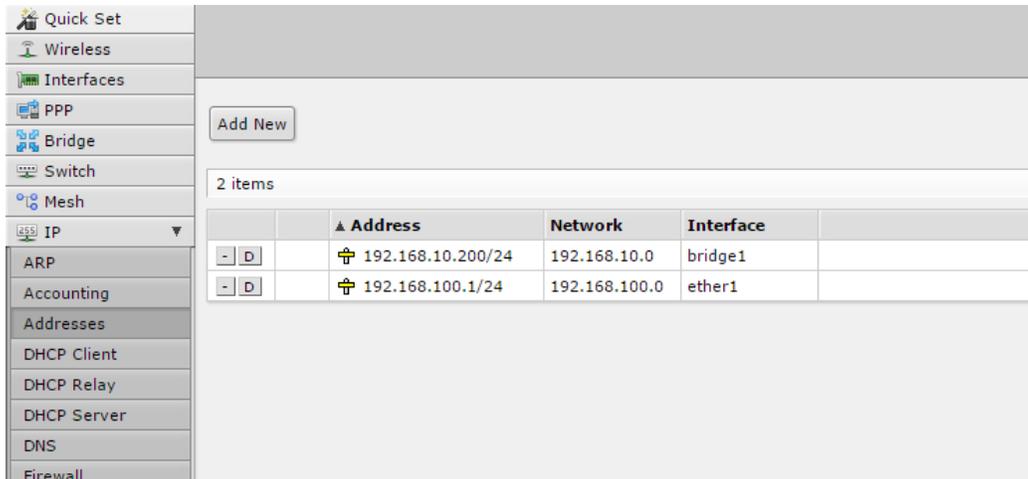


Imagen 13 Asignación de IP al bridge en Mikrotik

4.- Creación de flujos (Flows)

En la opción OpenFlow se debe crear un flujo asignando la dirección IP del controlador y el puerto del controlador, en la imagen 14 se puede observar este proceso.

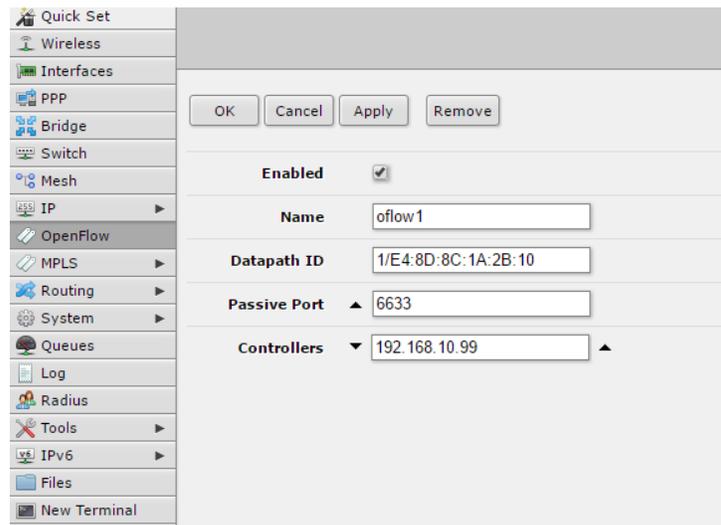
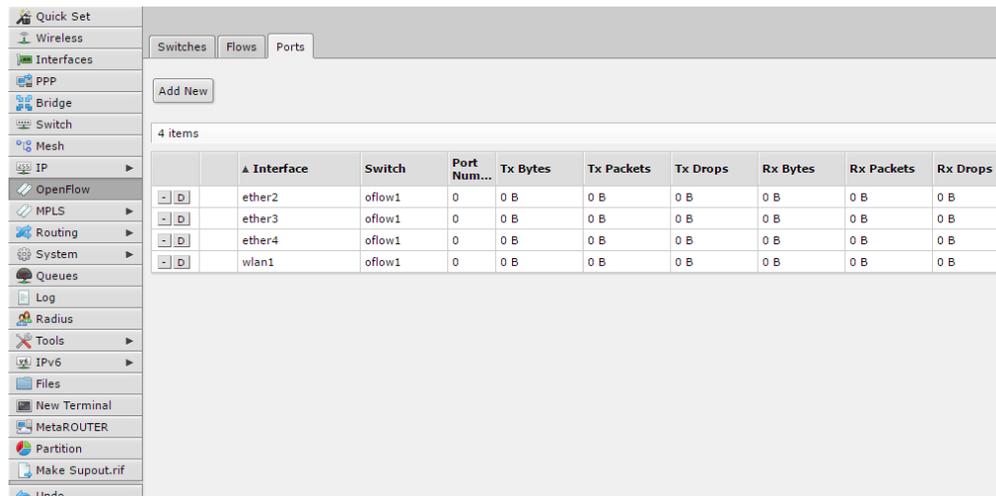


Imagen 14 Creación de flows en Mikrotik

5.- Asignar Puertos al Flujo creado

En la pestaña Ports se deben elegir los puertos que van a corresponder a la red SDN los mismos que utilizarán el protocolo OpenFlow.

En la imagen 15 se puede observar los puertos asignados.



The screenshot shows a network configuration tool with a sidebar on the left containing various configuration categories like Wireless, Interfaces, PPP, Bridge, Switch, Mesh, IP, OpenFlow, MPLS, Routing, System, Queues, Log, Radius, Tools, IPv6, Files, New Terminal, MetaROUTER, Partition, and Make Supout.rif. The main area has tabs for 'Switches', 'Flows', and 'Ports'. The 'Ports' tab is active, showing a table with 4 items. The table has columns for Interface, Switch, Port Num..., Tx Bytes, Tx Packets, Tx Drops, Rx Bytes, Rx Packets, and Rx Drops. The data rows are as follows:

	▲ Interface	Switch	Port Num...	Tx Bytes	Tx Packets	Tx Drops	Rx Bytes	Rx Packets	Rx Drops
- D	ether2	oflow1	0	0 B	0 B	0 B	0 B	0 B	0 B
- D	ether3	oflow1	0	0 B	0 B	0 B	0 B	0 B	0 B
- D	ether4	oflow1	0	0 B	0 B	0 B	0 B	0 B	0 B
- D	wlan1	oflow1	0	0 B	0 B	0 B	0 B	0 B	0 B

Imagen 15 Asignación de puertos al flow

Instalación de OpenStack.

Para la instalación de OpenStack se creó una máquina virtual en VMWare con las siguientes características:

- Sistema operativo Ubuntu 14.04 LTS.
- Memoria RAM de 6 GB.
- Disco duro de 100 GB.

1.- Actualización de repositorios:

Para iniciar con la instalación de OpenStack es necesario actualizar los repositorios de Ubuntu de la siguiente manera:

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

2.- Instalar git

El complemento git es necesario para clonar los paquetes necesarios durante la instalación de OpenStack.

```
sudo apt-get install git
```

3.- Clonación de OpenStack

A continuación se debe clonar OpenStack liberty con el siguiente comando.

```
git clone https://www.github.com/openstack-dev/devstack.git -b stable/liberty
```

4.- Configurar parámetros para integración de Neutrón con OpenStack

Para proceder a integrar Neutrón con OpenStack es necesario crear un archivo llamado “localrc” en la carpeta “devstack” con los siguientes parámetros:

```
disable_service n-net
```

```
enable_service q-svc q-agt q-dhcp q-l3
```

```
enable_service s-proxy s-object s-container s-account
```

5.- Iniciar Instalación

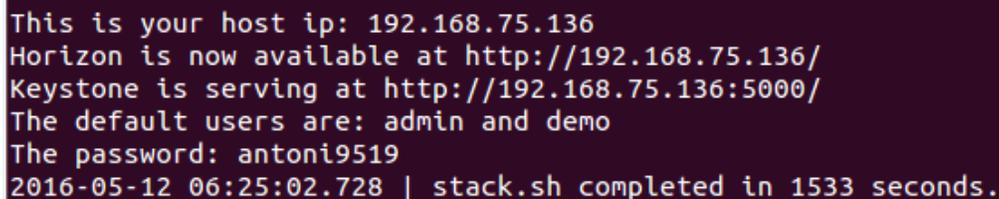
Una vez creado el archivo para la integración de Neutrón se debe ejecutar el archivo stack.sh con el siguiente comando:

```
./stack.sh
```

Al momento de realizar este paso empezara la instalación, se pedirá ingresar algunas contraseñas las mismas que sirven para la administración del OpenStack.

6.- Verificación de la instalación

Completada la instalación se mostrará la información resultante como en la imagen 16.



```
This is your host ip: 192.168.75.136
Horizon is now available at http://192.168.75.136/
Keystone is serving at http://192.168.75.136:5000/
The default users are: admin and demo
The password: antoni9519
2016-05-12 06:25:02.728 | stack.sh completed in 1533 seconds.
```

Imagen 16 IP del servidor y Credenciales para logueo en OpenStack

La primera línea corresponde a la dirección IP a la cual se debe ingresar para usar OpenStack y las líneas cuatro y cinco contienen el usuario y la contraseña para la administración del software.

7.- Ingresar al ambiente web

Para ingresar a OpenStack mediante un ambiente web es necesario ingresar la IP que fue dada al terminar la instalación, el usuario es “admin” y el password corresponde al ingresado en el momento de la instalación, la imagen de logueo de OpenStack es parecida a la imagen 17.

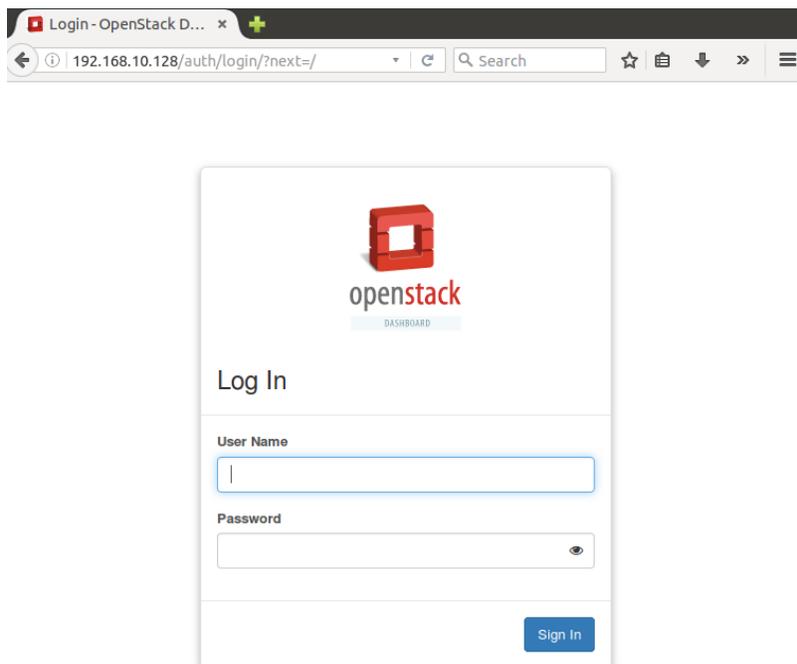


Imagen 17 Interfaz de logueo de OpenStack

Una vez ingresada las credenciales se debe verificar en el menú izquierdo de la interfaz como se muestra en la imagen 18 que se encuentre la opción “Network”, lo que nos indica que neutrón se instaló correctamente.

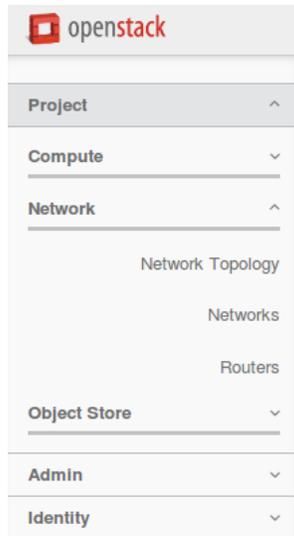


Imagen 18 Menú de OpenStack

Configuración de OpenStack

1.- Creación del router neutrón

En el apartado *Network / Routers* se debe proceder a crear un nuevo router.

Routers



Imagen 19 Creación de router Neutrón

Al seleccionar la opción “*Create Router*” se debe ingresar los siguientes parámetros:

- Un nombre para el Router
- Estado en el que se crea el Router.

Elegir una red externa para el router (por defecto se crea una red pública la misma que será usada para el router y permitirá la conexión con la red SDN).

En la imagen 20 se puede observar los campos solicitados.

Create Router x

Router Name *

Description:
 Creates a router with specified parameters.

Admin State

External Network

Imagen 20 Parámetros de creación de router

Una vez creado el enrutador se visualizará de una forma parecida a la imagen 21:

Routers

	Name	Status	External Network	Admin State	Actions
<input type="checkbox"/>	Router	Active	-	UP	Set Gateway ▾

Displaying 1 item

Imagen 21 Listado de routers en OpenStack

Esto indica que el enrutador que funcionara como middleware fue creado correctamente.

2.- Creación de la red local

A continuación se debe crear una red local, la misma que servirá para poder conectar las instancias tanto en la red interna como con la red pública. Para esto es necesario seleccionar el menú *Network/Networks* y seguidamente elegir la opción “*Create Network*”

Networks

Imagen 22 Opción de creación de redes en OpenStack

De igual manera se pedirán algunos parámetros para la creación de la red local como se observa en la imagen 23.

Create Network x

NetworkSubnet *Subnet Details

Network Name

Create a new network. In addition, a subnet associated with the network can be created in the next panel.

Admin State * ?

Imagen 23 Parámetros para la creación de una red en OpenStack

3.- Creación de una Subred

Lo siguiente será configurar una subred, para esto se debe elegir un nombre, la dirección de Red, la versión del protocolo de internet y definir un Gateway para la red, todos estos parámetros deben ser ingresados en la interfaz que se muestra en la imagen 24.

Create Network x

NetworkSubnetSubnet Details

Create Subnet

Create a subnet associated with the new network, in which case "Network Address" must be specified. If you wish to create a network without a subnet, uncheck the "Create Subnet" checkbox.

Subnet Name

Network Address * ?

IP Version *

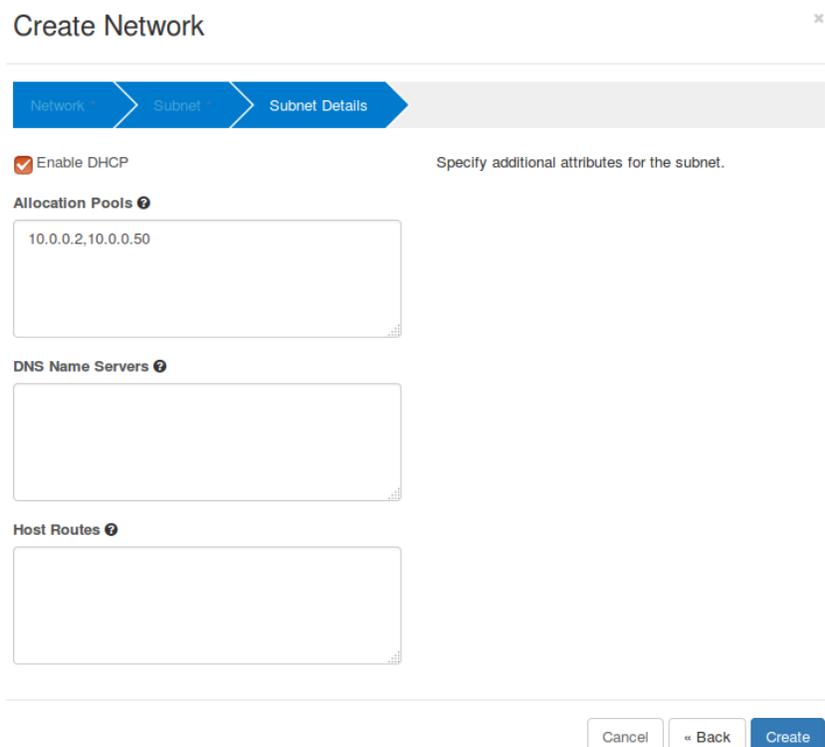
Gateway IP ?

Disable Gateway

Imagen 24 Parámetros para la creación de una red en OpenStack

4.- Creación de un pool DHCP

Después se debe ingresar el rango de IP que serán utilizadas para asignar dichas direcciones a las distintas instancias instaladas en OpenStack.



Create Network

Network > Subnet > Subnet Details

Enable DHCP Specify additional attributes for the subnet.

Allocation Pools ?

10.0.0.2,10.0.0.50

DNS Name Servers ?

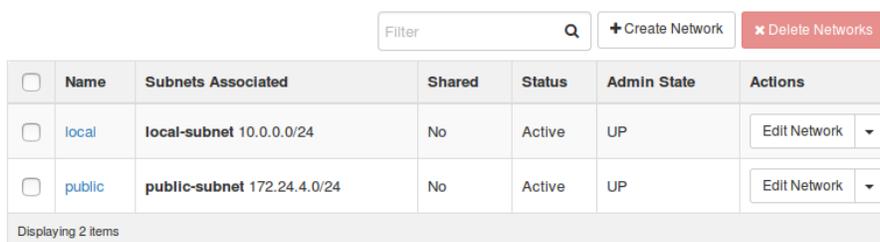
Host Routes ?

Cancel Back Create

Imagen 25 Parámetros para la creación de una red en OpenStack

Al finalizar todos los pasos se tendrá una interfaz parecida a la imagen 26 con todas las redes creadas en este caso una pública y una local:

Networks



<input type="checkbox"/>	Name	Subnets Associated	Shared	Status	Admin State	Actions
<input type="checkbox"/>	local	local-subnet 10.0.0.0/24	No	Active	UP	Edit Network
<input type="checkbox"/>	public	public-subnet 172.24.4.0/24	No	Active	UP	Edit Network

Displaying 2 items

Imagen 26 Listado de redes existentes en OpenStack

5.- Subir imagen de Ubuntu

A continuación se debe subir una imagen previamente configurada del sistema operativo Ubuntu 14, la misma que servirá para crear la instancia y sobre la cual se instalará el servidor

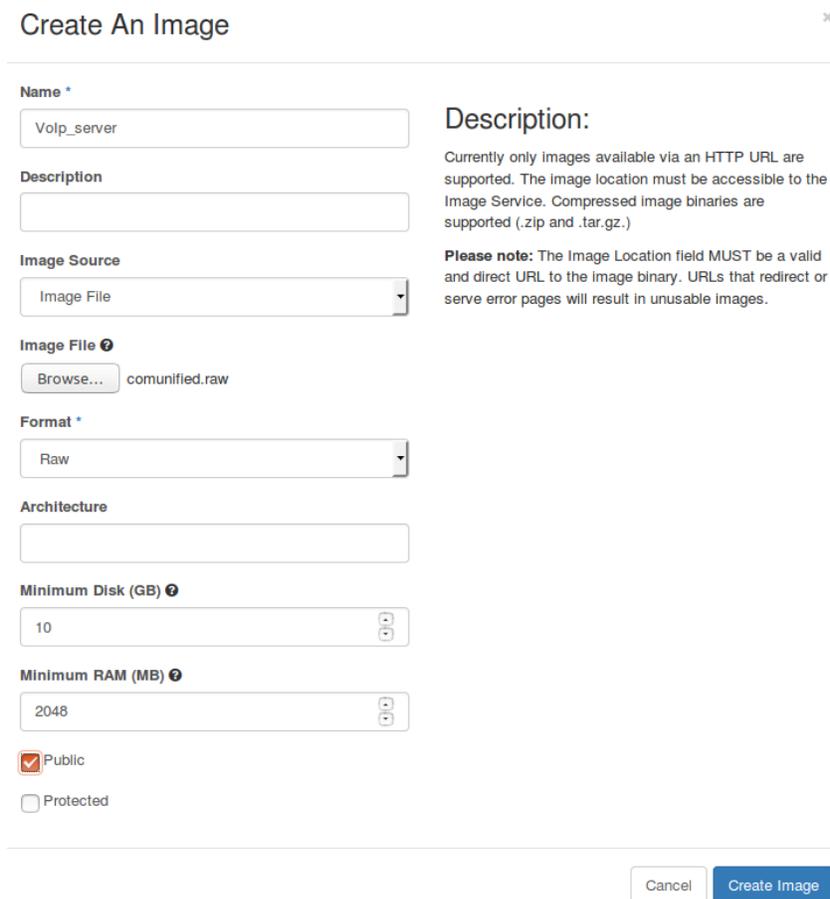
de VoIP. Para esto es necesario seleccionar el menú “*Compute/Images*” y elegir la opción “*Create Image*”.

Images



Imagen 27 Creación de imágenes en OpenStack

Una vez seleccionada la opción para crear la imagen será necesario ingresar los parámetros siguientes: Un nombre de la imagen, una descripción, ubicación de la imagen, formato de la imagen, espacio mínimo de disco, capacidad mínima de RAM como se puede observar en la imagen 28.



Create An Image ×

Name *
Volp_server

Description

Image Source
Image File

Image File ⓘ
Browse... comunified.raw

Format *
Raw

Architecture

Minimum Disk (GB) ⓘ
10

Minimum RAM (MB) ⓘ
2048

Public
 Protected

Description:
Currently only images available via an HTTP URL are supported. The image location must be accessible to the Image Service. Compressed image binaries are supported (.zip and .tar.gz.)

Please note: The Image Location field MUST be a valid and direct URL to the image binary. URLs that redirect or serve error pages will result in unusable images.

Cancel Create Image

Imagen 28 Parámetros de creación de imágenes en OpenStack

Luego de crear la imagen, esta aparecerá en el listado de imágenes de OpenStack.

Images

<input type="checkbox"/>	Image Name	Type	Status	Public	Protected	Format	Size	Actions
<input type="checkbox"/>	Voip-image	Image	Active	Yes	No	Raw	1.9 GB	Launch Instance ▾
<input type="checkbox"/>	cirros-0.3.4-x86_64-uec	Image	Active	Yes	No	AMI	24.0 MB	Launch Instance ▾
<input type="checkbox"/>	cirros-0.3.4-x86_64-uec-ramdisk	Image	Active	Yes	No	ARI	3.6 MB	Edit Image ▾
<input type="checkbox"/>	cirros-0.3.4-x86_64-uec-kernel	Image	Active	Yes	No	AKI	4.7 MB	Edit Image ▾

Displaying 4 items

Imagen 29 Listado de imágenes existentes en OpenStack

6.- Creación de llaves para las instancias

Para crear la instancia primero se debe crear un par de llaves para el acceso mediante ssh, esto se lo hace seleccionando la opción “*Compute/ Access & Security/ Key pairs*” en el menú, finalmente se debe ingresar un nombre y seleccionar el botón “*Create Key Pair*”.

Create Key Pair ✕

Key Pair Name *

Description:

Key pairs are ssh credentials which are injected into images when they are launched. Creating a new key pair registers the public key and downloads the private key (a .pem file).

Protect and use the key as you would any normal ssh private key.

Imagen 30 Creación de un par de llaves en OpenStack

El resultado será parecido a la imagen 31.

<input type="checkbox"/>	Key Pair Name	Fingerprint	Actions
<input type="checkbox"/>	final_voip	ee:a4:eb:86:1c:6d:1c:65:c8:00:4b:1d:ee:b8:27:92	Delete Key Pair

Displaying 1 item

Imagen 31 Listado de llaves existentes en OpenStack

7.- Creación de políticas de seguridad

Las políticas de seguridad son necesarias para acceder a las instancias e incluso para que los servicios funcionen correctamente, esto se lo hace seleccionando la opción “*compute/ Access & Security/ Security Groups*”, a continuación se debe elegir “*manage rules*” para crear nuevas políticas.

<input type="checkbox"/>	Name	Description	Actions
<input type="checkbox"/>	default	Default security group	Manage Rules

Displaying 1 item

Imagen 32 Opción de creación de políticas de seguridad para instancias de OpenStack

Luego se debe agregar una nueva regla en “*Add Rule*”, en este apartado se definen las reglas necesarias para la instancia. En la imagen 33 se puede observar este procedimiento.

Add Rule

Rule *
All ICMP

Direction
Ingress

Remote * ⓘ
CIDR

CIDR ⓘ
0.0.0.0/0

Description:
Rules define which traffic is allowed to instances assigned to the security group. A security group rule consists of three main parts:
Rule: You can specify the desired rule template or use custom rules, the options are Custom TCP Rule, Custom UDP Rule, or Custom ICMP Rule.
Open Port/Port Range: For TCP and UDP rules you may choose to open either a single port or a range of ports. Selecting the “Port Range” option will provide you with space to provide both the starting and ending ports for the range. For ICMP rules you instead specify an ICMP type and code in the spaces provided.
Remote: You must specify the source of the traffic to be allowed via this rule. You may do so either in the form of an IP address block (CIDR) or via a source group (Security Group). Selecting a security group as the source will allow any other instance in that security group access to any other instance via this rule.

Add

Imagen 33 Parámetros de creación de políticas de seguridad

De la misma manera se debe proceder a crear todas las reglas necesarias.

Manage Security Group Rules: default (02fc5419-86bc-4afc-97ef-f174e1ab4a96)

<input type="checkbox"/>	Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Actions
<input type="checkbox"/>	Egress	IPv6	Any	Any	:::0	-	Delete Rule
<input type="checkbox"/>	Egress	IPv4	Any	Any	0.0.0.0/0	-	Delete Rule
<input type="checkbox"/>	Ingress	IPv6	Any	Any	-	default	Delete Rule
<input type="checkbox"/>	Ingress	IPv4	Any	Any	-	default	Delete Rule
<input type="checkbox"/>	Ingress	IPv4	ICMP	Any	0.0.0.0/0	-	Delete Rule
<input type="checkbox"/>	Egress	IPv4	ICMP	Any	0.0.0.0/0	-	Delete Rule
<input type="checkbox"/>	Egress	IPv4	ICMP	Any	-	default	Delete Rule
<input type="checkbox"/>	Ingress	IPv4	ICMP	Any	-	default	Delete Rule
<input type="checkbox"/>	Ingress	IPv4	TCP	1 - 65535	0.0.0.0/0	-	Delete Rule
<input type="checkbox"/>	Egress	IPv4	TCP	1 - 65535	0.0.0.0/0	-	Delete Rule
<input type="checkbox"/>	Ingress	IPv4	TCP	22 (SSH)	0.0.0.0/0	-	Delete Rule
<input type="checkbox"/>	Egress	IPv4	UDP	1 - 65535	0.0.0.0/0	-	Delete Rule
<input type="checkbox"/>	Ingress	IPv4	UDP	1 - 65535	0.0.0.0/0	-	Delete Rule

Imagen 34 Listado de políticas de seguridad creadas

8.- Creación de la instancia

Una vez que se crearon las llaves y las políticas necesarias se debe proceder a crear la instancia a partir de la imagen ya configurada, para esto es necesario ingresar a las imágenes de OpenStack y seleccionar la opción “*Launch Instance*”.

<input type="checkbox"/>	Image Name	Type	Status	Public	Protected	Format	Size	Actions
<input type="checkbox"/>	Voip-image	Image	Active	Yes	No	Raw	1.9 GB	Launch Instance

Imagen 35 Opción para la creación de un instancia a partir de una imagen

Como en pasos anteriores, se solicitarán diferentes parámetros para crear la instancia, primero es necesario definir los detalles, nombre de la instancia, el sabor (Las características de CPU, RAM disco que se dará a la instancia).

En la imagen 36 se observa estos campos que son necesarios para el correcto funcionamiento de la instancia.

Launch Instance

Details * Access & Security Networking * Post-Creation Advanced Options

Availability Zone
nova

Instance Name *
Serificio_Voip

Flavor * ?
m1.small
Some flavors not meeting minimum image requirements have been disabled.

Instance Count * ?
1

Instance Boot Source * ?
Boot from image

Image Name *
Voip-image (1.9 GB)

Specify the details for launching an instance.
The chart below shows the resources used by this project in relation to the project's quotas.

Flavor Details

Name	m1.small
VCPUs	1
Root Disk	20 GB
Ephemeral Disk	0 GB
Total Disk	20 GB
RAM	2,048 MB

Project Limits

Number of Instances 1 of 10 Used
Number of VCPUs 1 of 20 Used
Total RAM 2,048 of 51,200 MB Used

Cancel Launch

Imagen 36 Parámetros para la creación de una instancia

En el apartado de Access & Security se debe seleccionar la llave creada, y el grupo de seguridad.

Launch Instance

Details * Access & Security Networking * Post-Creation Advanced Options

Key Pair ?
final_voip

Security Groups ?
 default

Control access to your instance via key pairs, security groups, and other mechanisms.

Cancel Launch

Imagen 37 Parámetros para la creación de una instancia

En la parte de networking se debe elegir la red a la que pertenecerá la instancia, en este caso una red local.

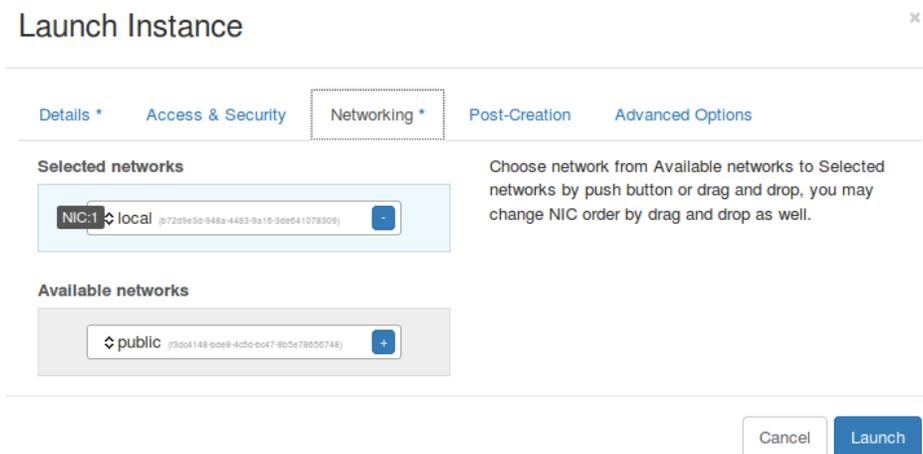


Imagen 38 Parámetros para la creación de una instancia

Una vez creada y ejecutada la instancia, aparecerá una ventana parecida a la imagen 39 en la cual se observa que todo funciona correctamente, es importante observar la columna “Power State”:

<input type="checkbox"/>	Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
<input type="checkbox"/>	Servicio-de-Volp	Voip-image	10.0.0.5	m1.small	final_volp	Active	nova	None	Running	4 days, 23 hours	Create Snapshot

Displaying 1 item

Imagen 39 Listado de las instancias creadas en OpenStack

9.- Verificación de la red creada

En el apartado de Network se podrá observar la topología creada a partir del router Neutron y la instancia generada, así como también se podrá ver la red pública a la cual se conectarán los dispositivos de la red SDN. En la imagen 40 se muestra la topología de OpenStack en la que se visualiza Neutrón (enrutador), la instancia que está representada por una computadora y la red pública que se conecta directamente al enrutador.

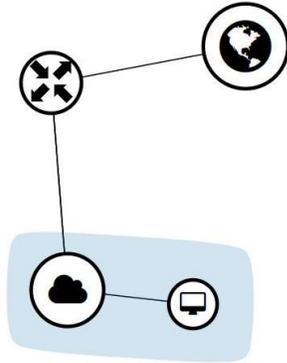


Imagen 40 Topología de las instancias y Neutron en OpenStack

Al ingresar el comando “*ifconfig*” se podrá observar una interfaz llamada “*br-ex*” que pertenece al router neutrón y a la cual se conectarán los dispositivos de la red externa.

```
br-ex  Link encap:Ethernet  HWaddr 3a:04:5b:39:e3:44
      inet addr:172.24.4.1  Bcast:172.24.4.255  Mask:255.255.255.0
      inet6 addr: fe80::6c14:26ff:fec2:cdfc/64 Scope:Link
      UP BROADCAST RUNNING MTU:1500 Metric:1
      RX packets:6034 errors:0 dropped:0 overruns:0 frame:0
      TX packets:5127 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:1584295 (1.5 MB)  TX bytes:624382 (624.3 KB)
```

Imagen 41 Verificación de la interfaz *br-ex* perteneciente a Neutrón

10.- Creación de rutas

Para acceder a la instancia de Ubuntu es necesario ingresar las rutas que se observan en la imagen 42.

```
openstack@ubuntu:~/devstack$ route
Kernel IP routing table
Destination      Gateway         Genmask        Flags Metric Ref    Use Iface
default          192.168.10.1   0.0.0.0        UG    0     0      0 eth0
10.0.0.0         172.24.4.2    255.255.255.0  UG    0     0      0 br-ex
172.24.4.0       *              255.255.255.0  U     0     0      0 br-ex
192.168.10.0     *              255.255.255.0  U     1     0      0 eth0
192.168.10.0     *              255.255.255.0  U     1     0      0 eth1
192.168.122.0    *              255.255.255.0  U     0     0      0 virbr0
```

Imagen 42 Rutas creadas en Ubuntu para acceso a OpenStack

Integrar OpenStack - Neutrón con el controlador OpenDayLight

Para integrar OpenStack con el controlador OpenDayLight es necesario crear otra interfaz de red para la máquina virtual en la cual se encuentra instalado Ubuntu y OpenStack, donde la primera interfaz servirá para permitir la conexión de los flujos de datos con el protocolo OpenFlow y la otra para lo conexión con el controlador.

1.- Reiniciar el servicio openvswitch

El primer paso es detener y levantar el servicio de openvswitch con los siguientes comandos

```
sudo service openvswitch-switch stop
```

```
sudo service openvswitch-switch start
```

2.- Configuración del puente

Luego se debe crear un puente hacia la red interna de OpenStack mediante la segunda interfaz física de Ubuntu con el siguiente comando:

```
sudo ovs-vsctl add-port br-int eth1
```

En donde “br-int” es el nombre de la interfaz interna de neutrón, “eth1” es el nombre de la interfaz física de la máquina virtual de Ubuntu.

3.- Crear conexión para el controlador ODL

Por último se debe a crear la conexión con el controlador OpenDayLight de la siguiente manera:

```
sudo ovs-vsctl set-manager tcp:192.168.10.99:6640
```

Donde “192.168.10.99” es la dirección IP del controlador, “6640” es el puerto por donde se comunicará Neutrón con el controlador.

Un vez realizado este paso se podrá verificar su correcta configuración con el siguiente comando:

```
sudo ovs-vsctl show
```

Obteniendo un resultado como el de la imagen 43.

```
openstack@ubuntu:~$ sudo ovs-vsctl show
4003762b-db92-4281-bbdd-43f3c5a00223
  Manager "tcp:192.168.10.99:6640"
    is_connected: true
  Bridge br-int
    Controller "tcp:192.168.10.99:6653"
      is_connected: true
    fail_mode: secure
```

Imagen 43 Estado de conexión de Neutrón con ODL

En el controlador una vez agregado el módulo de openvswitch aparecerá la topología con los dispositivos tanto físicos como virtuales, la interfaz de ODL será parecida a la mostrada en la figura 44.

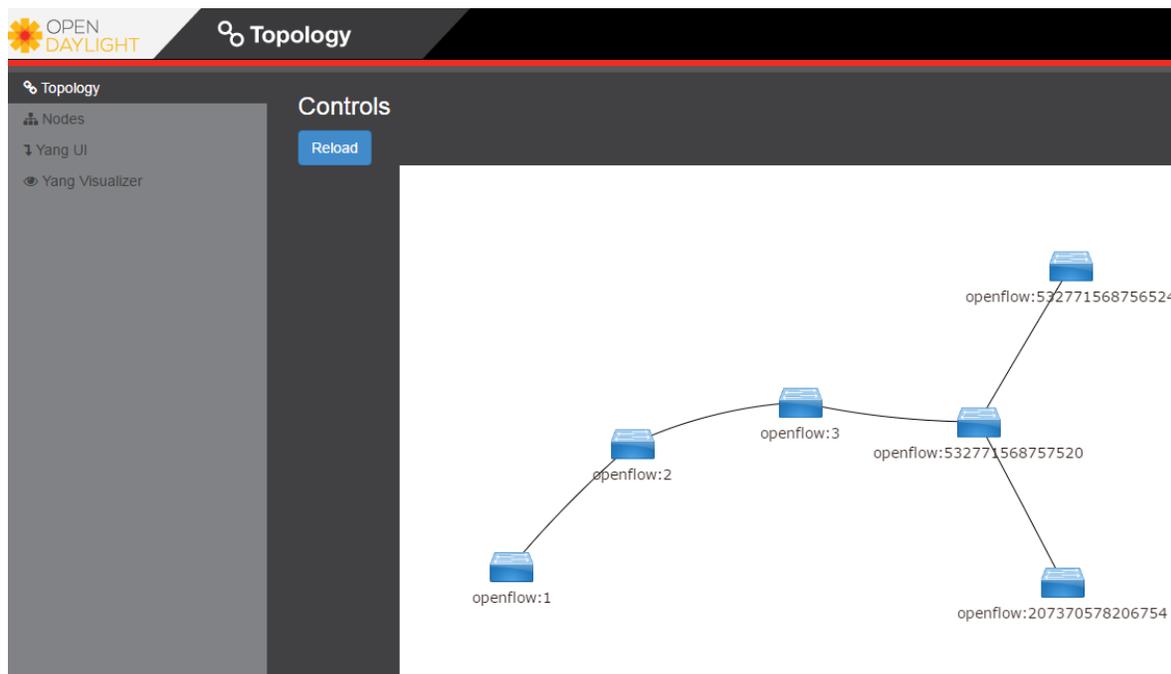


Imagen 44 Topología general del proyecto en ODL

En el diagrama se pueden visualizar seis conmutadores, estos pertenecen tanto a la red física como a la virtualizada en Mininet, los dispositivos openflow:1, openflow:2 y openflow:3 son los virtualizados, mientras que el dispositivo openflow:532771568757520 corresponde al dispositivo Mikrotik SDN2 y openflow:532771568756524 es el Mikrotik SDN1. Finalmente el elemento openflow:61921237922407 corresponde a neutrón.

Configuración de flujos para administrar el tráfico con ODL y conmutadores

Para configurar cada uno de los dispositivos es necesario conocer el identificador de cada uno de ellos, para esto se debe ingresar a la opción “Nodes” del menú y posteriormente aparecerá un listado con todos los dispositivos de la red parecida a la imagen siguiente:

Node Id	Node Name	Node Connectors
No data found		
openflow:532771568757520		5

Imagen 45 Listado de Dispositivos de la red

Una vez identificados los nodos de la red se puede proceder a configurarlos, para esto es necesario ingresar a la opción “Yang UI” al siguiente apartado:

Opendaylight-inventory / config / nodes / node {id} / table {id} / flow {id}

En la siguiente imagen se puede observar las opciones seleccionadas:

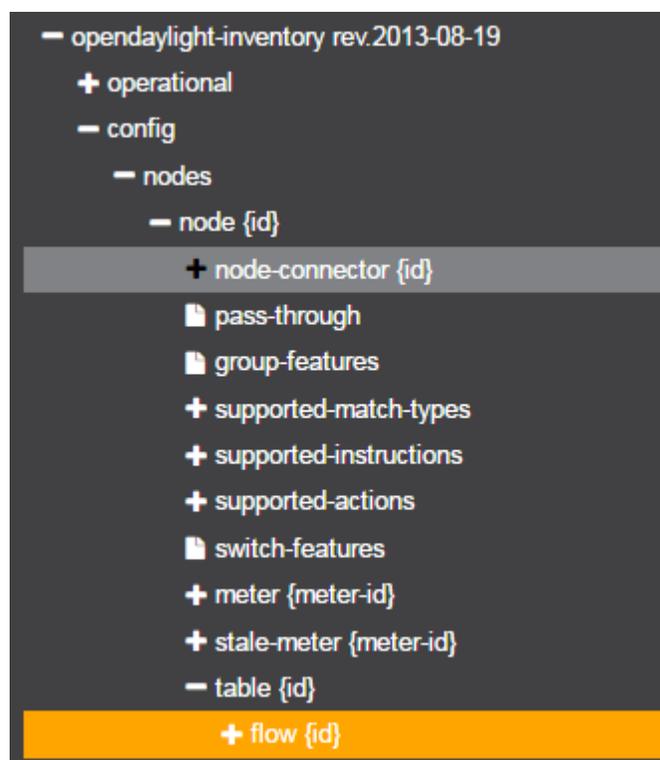


Imagen 46 Módulos de configuración de ODL

A continuación en la parte inferior de la interfaz aparecerán los campos necesarios a ingresar para configurar la red, en la imagen siguiente se presenta un ejemplo con los campos que se deben llenar para seleccionar el nodo a configurar así como también la tabla de flujos que se modificará y el identificador del flujo.



Imagen 47 Métodos de envío de configuración de ODL

Es necesario también observar en la imagen anterior que para ingresar un flujo a un dispositivo la configuración debe estar seleccionada con la opción “*PUT*”, para borrar un flujo la opción a elegir es “*DELETE*” y para extraer o llamar a un flujo existente la opción elegida debe ser “*GET*”.

Para enviar los flujos de configuración a los conmutadores es necesario llenar los parámetros que se observan en la siguiente imagen.

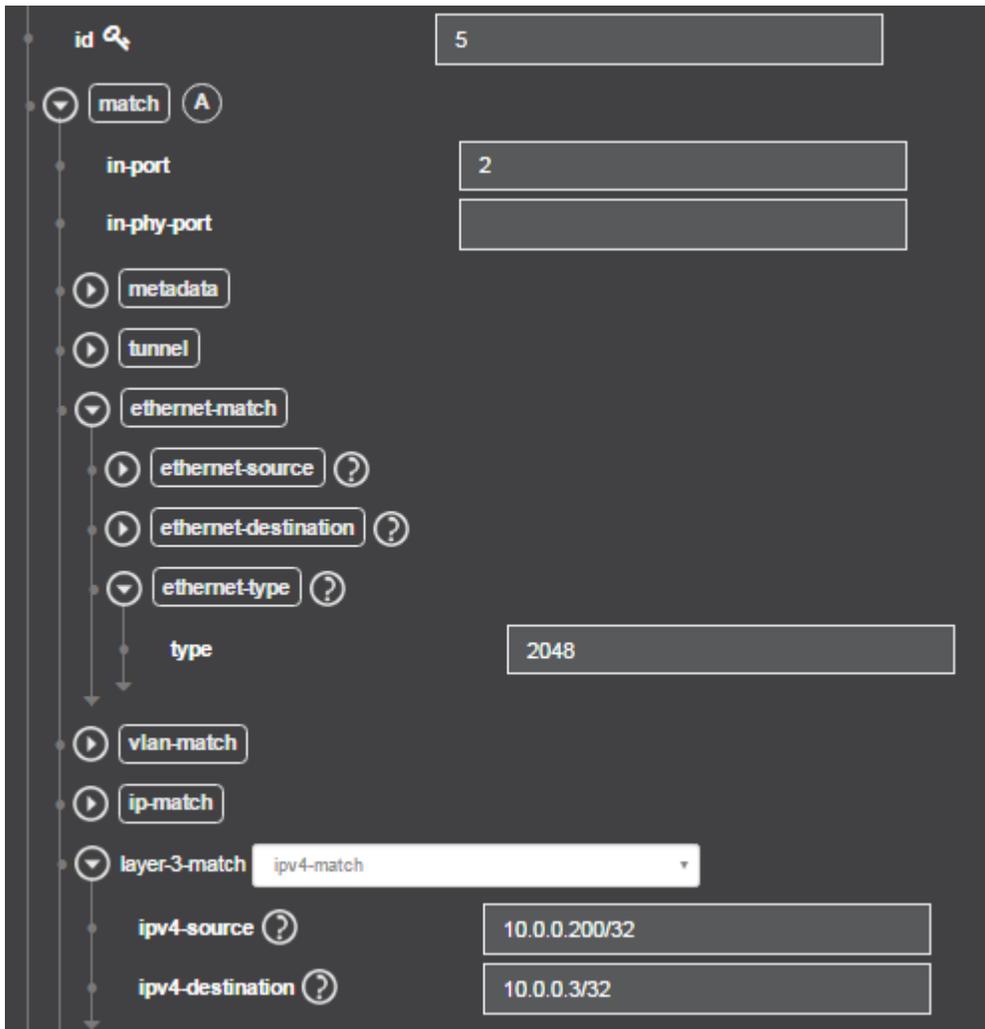


Imagen 48 Interfaz de configuración de ODL

- In-port.- Se especifica el puerto de entrada
- Ethernet-type.- indica el tamaño de las tramas
- Ipv4-sourse.- indica la dirección Ip de origen de los paquetes
- Ipv4-destination.- indica la dirección ip de destino de los paquetes

También es necesario crear instrucciones para indicar por cual es el puerto de salida de los paquetes para llegar al destino correcto, como muestra la siguiente imagen.

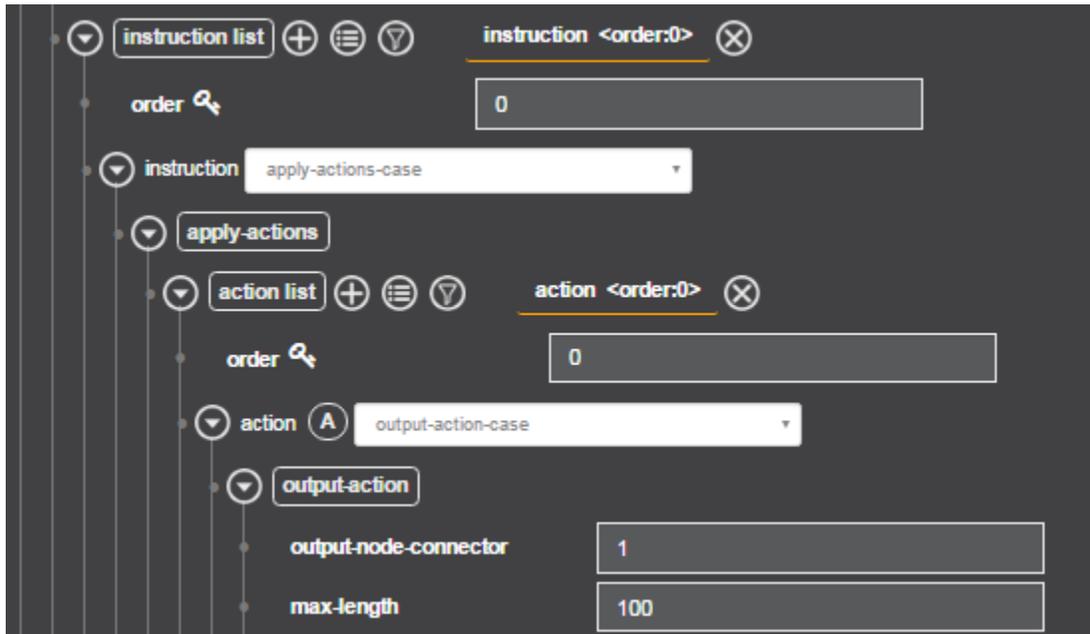


Imagen 49 Configuración de instrucciones de ODL

- Instrucción.- se especifica una acción para los flujos
- Action.- Indica la acción de salida
- Out-put-node-conector.- indica el puerto correcto de salida de los paquetes

También es necesario ingresar algunos parámetros como duración del flujo, prioridades id de la tabla como muestra la siguiente imagen

El campo prioridad indica la QoS que se aplica a los paquetes.

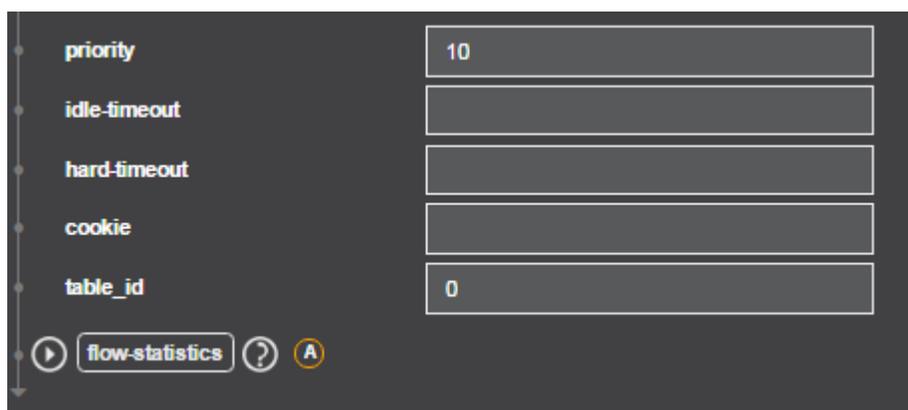


Imagen 50 Configuración de prioridad

Configuración de MPLS

Para configurar MPLS en los conmutadores de Mininet, es necesario agregar dos nuevas instrucciones la primera es llamada “*push-mpls-action-case*” que sirve para agregar la etiqueta a los paquetes al momento de ingresar a la red, para esto se ingresa el valor “*0x8847*” que corresponde al valor del protocolo a utilizar en este caso MPLS.

La segunda instrucción a crear es “*pop-mpls-action-case*”, la función de esta opción es la de quitar la etiqueta a los paquetes en el momento de salir de la red MPLS, para esto se debe ingresar el valor “*0x800*”, el mismo que corresponde al valor del protocolo IP versión 4.

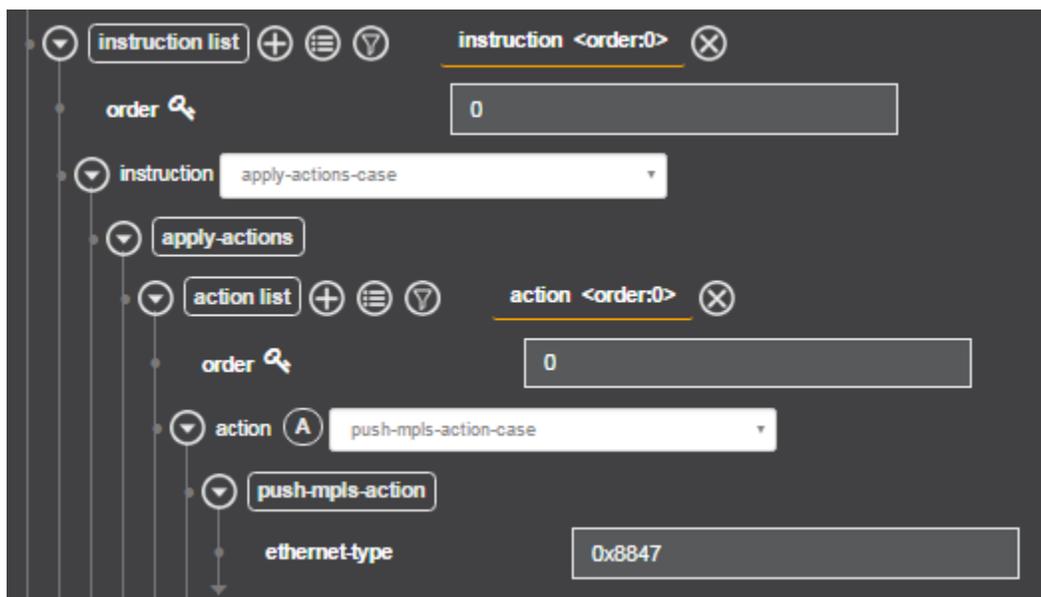


Imagen 51 Configuración de Push-MPLS

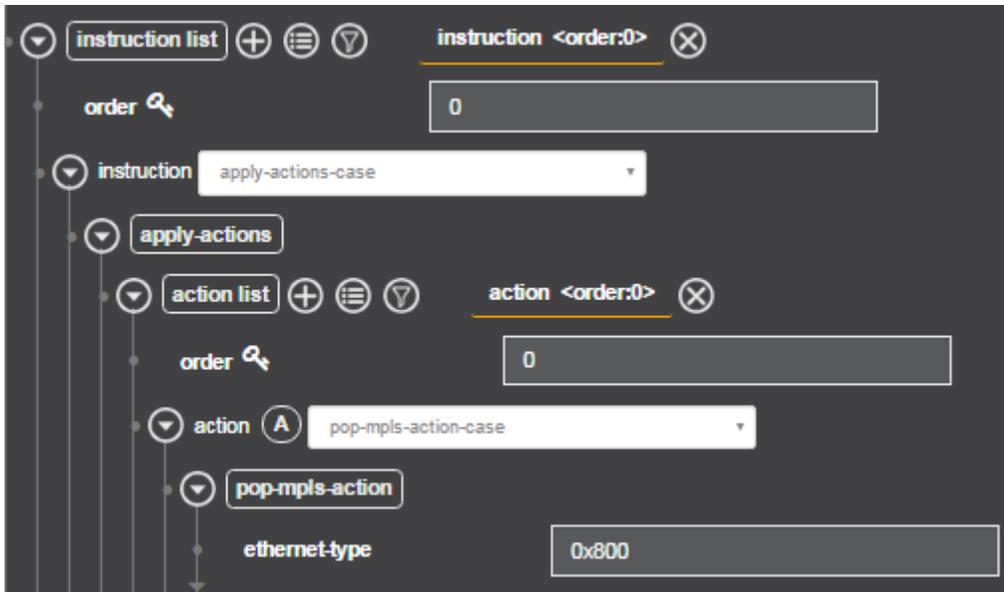


Imagen 52 Configuración de pop-MPLS

Finalmente en la opción “*protocol-match-fields*” se ingresa el valor de la etiqueta que serán asignados a los paquetes que ingresen al conmutador configurado.



Imagen 53 Configuración de etiqueta MPLS

RESULTADOS

Con la infraestructura de la plataforma terminada y realizadas las pruebas, los análisis que realizamos son:

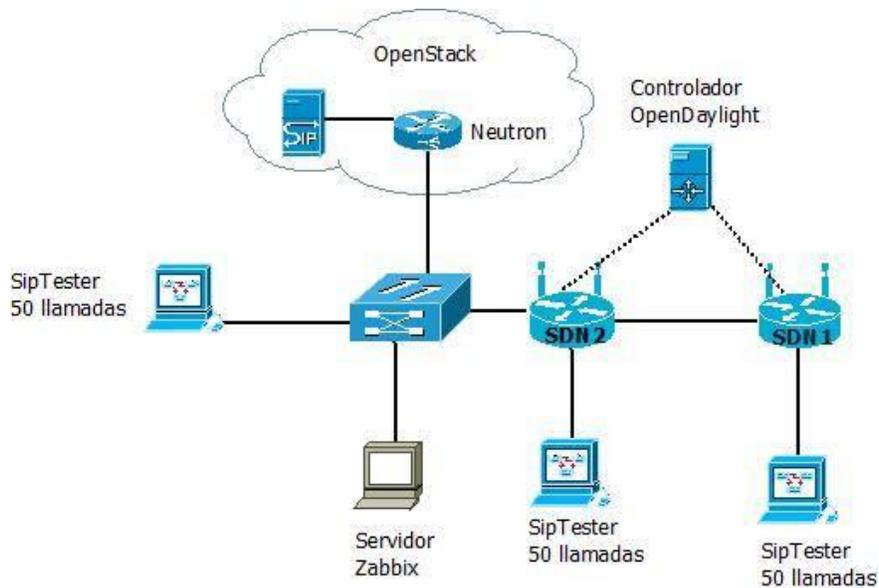


Imagen 54 Topología de pruebas

Rendimiento en el servidor VoIP

La primera evaluación de la plataforma fue una prueba de estrés en la que se realizaron 150 llamadas concurrentes. Utilizando la herramienta SipTester cuyas funcionalidades permite realizar ensayos de llamadas dinámicas desde diferentes puntos de la red SDN.

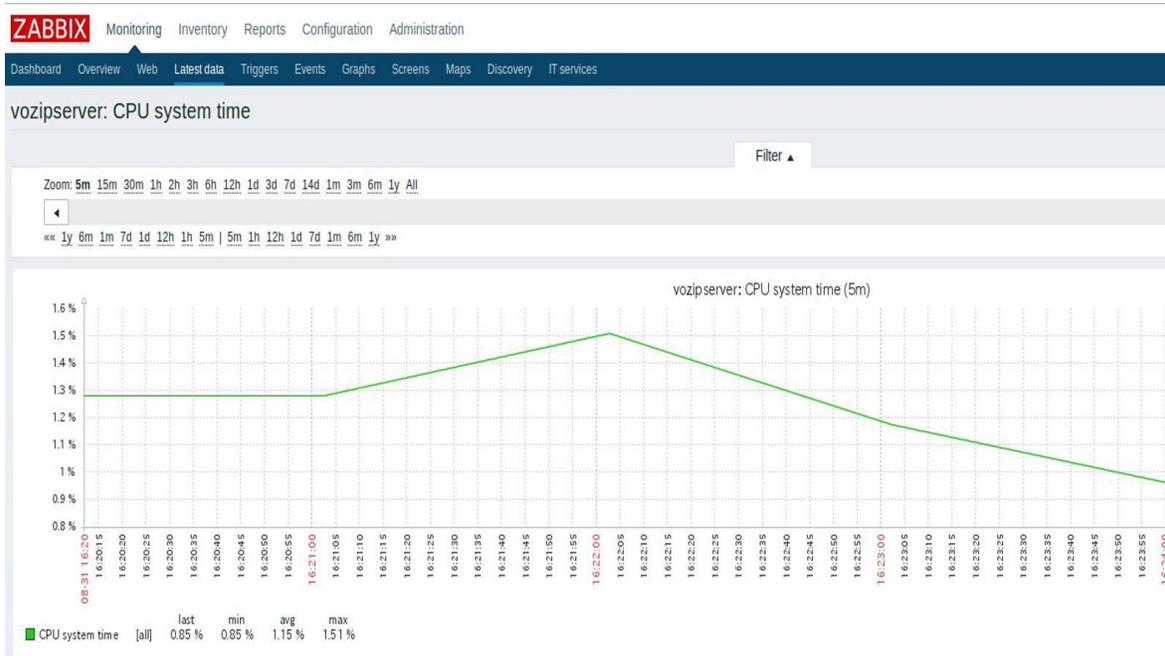


Imagen 55 Rendimiento del servidor de VoIP

En la imagen observamos el uso del CPU al momento de realizar la prueba de 150 llamadas y se encuentra en 1.5% por lo que nuestro servidor tiene suficiente capacidad.

Rendimiento de OpenStack.

En la siguiente imagen muestra el uso del CPU en la nube OpenStack, podemos observar que el uso máximo de CPU es de 5%

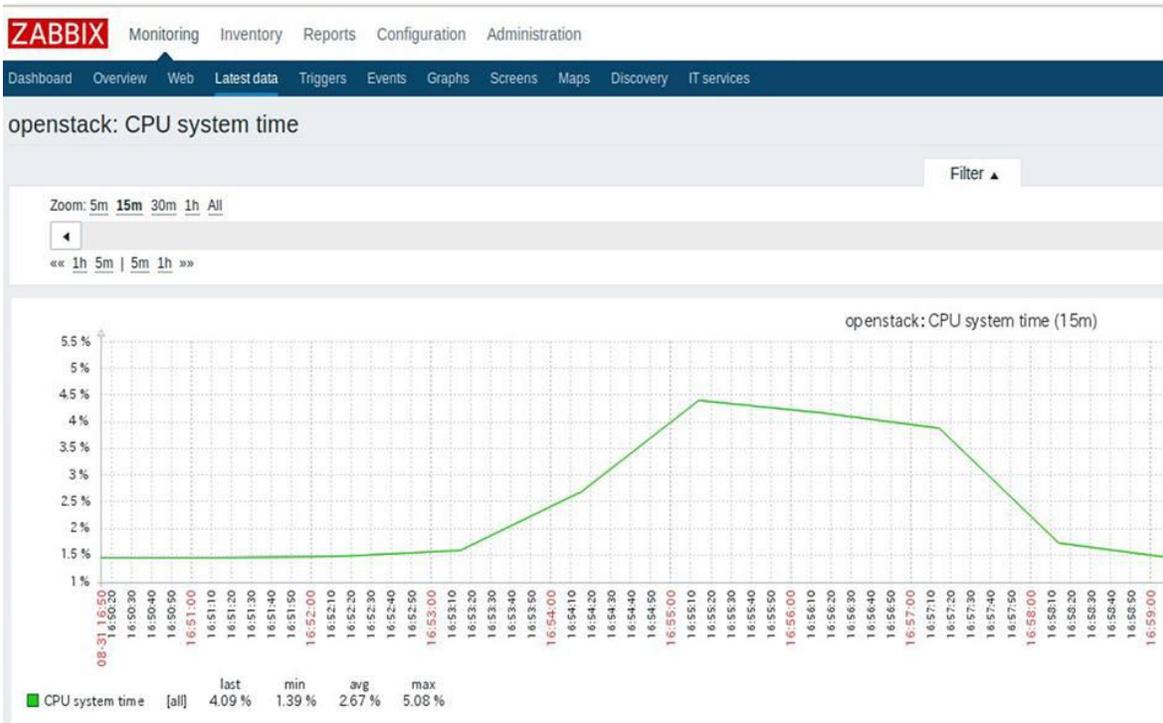


Imagen 56 Rendimiento del CPU de OpenStack

Tráfico de datos en el servidor Voip

La imagen muestra el tráfico generado por las 150 llamadas concurrentes ocupando un ancho de banda de 1.1 kbps.

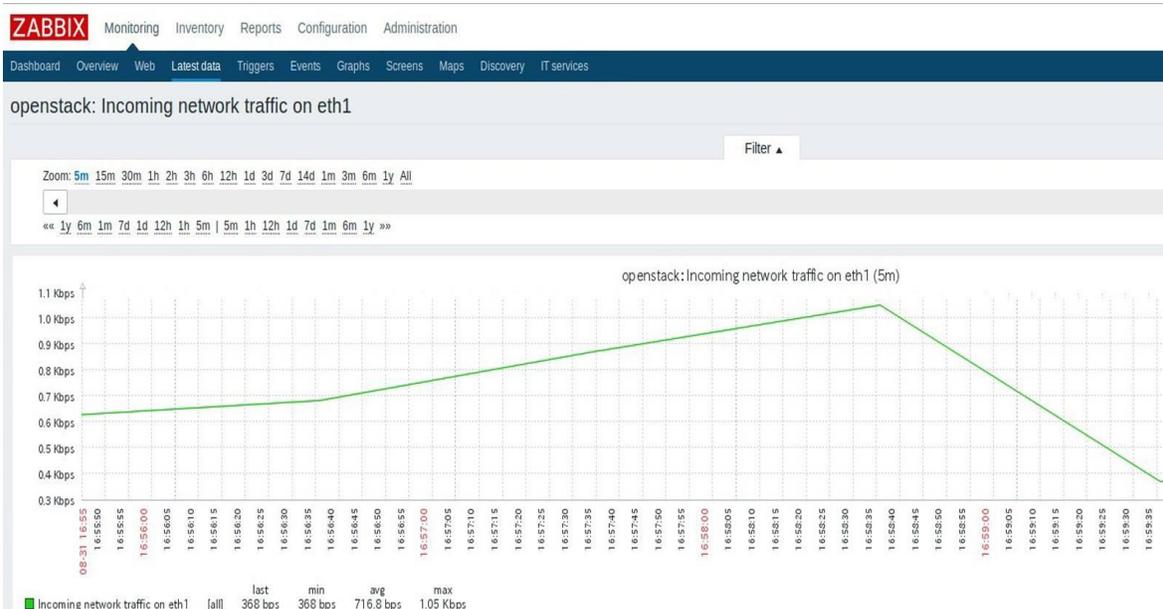


Imagen 57 Tráfico de la interfaz de red en el servidor de VoIP

Tráfico de datos en la nube OpenStack

En la nube OpenStack ocupa un ancho de banda de 10.12 kbps

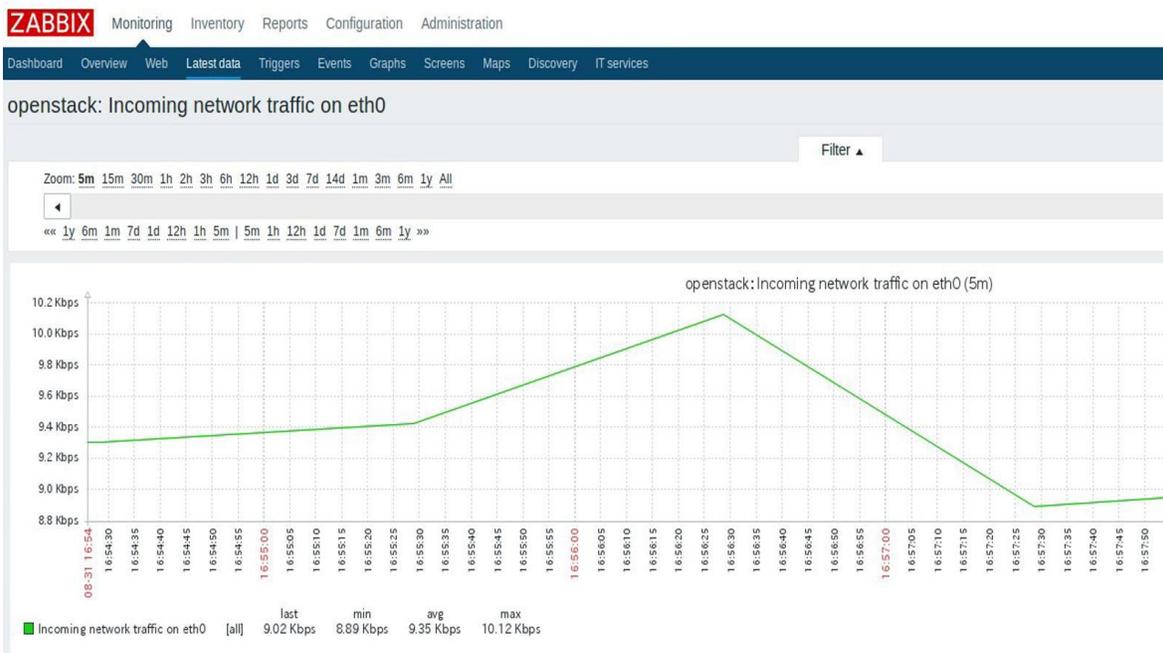


Imagen 58 Tráfico de la interfaz de red en OpenStack

Pruebas de MPLS en SDN

La configuración de MPLS se realizó virtualizando el plano de datos con Mininet en el cual se realizó un análisis con Wireshark que muestra los paquetes etiquetados con el protocolo MPLS, la comunicación se realiza entre el servidor de voz IP ubicado en OpenStack y un usuario del conmutador s1 de Mininet.

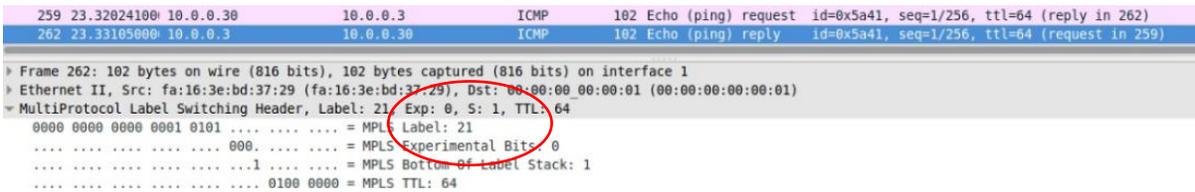


Imagen 59 Verificación de MPLS en Wireshark

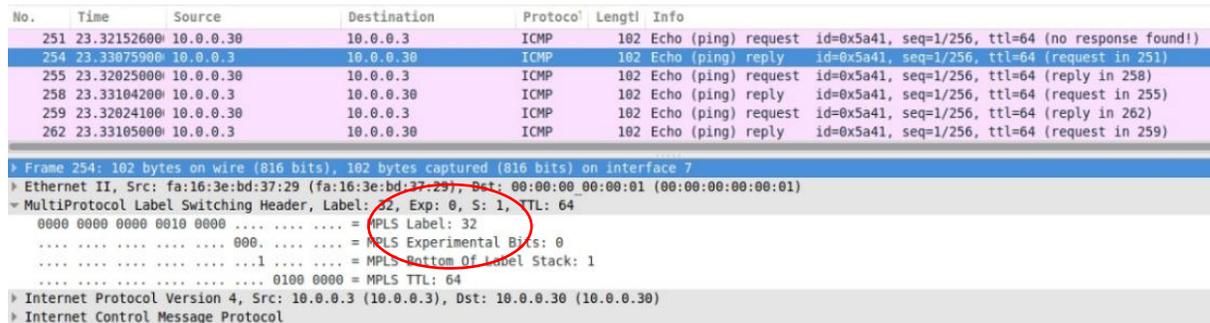


Imagen 60 Verificación de MPLS en Wireshark

También se realizó un monitoreo del tráfico de los dispositivos Mikrotik para verificar el tráfico OpenFlow que pasa por el dispositivo al realizar 150 llamadas concurrentes durante 30 segundos.

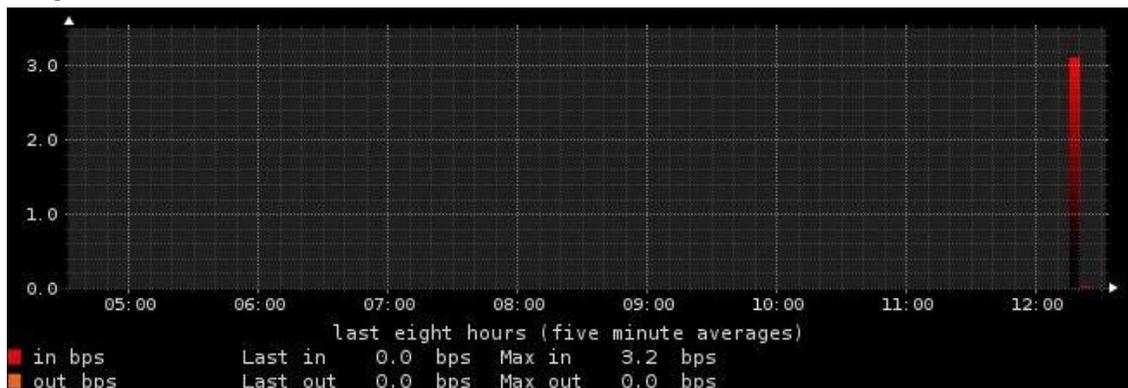


Imagen 61 Tráfico Openflow en Mikrotik

Prueba de calidad de servicio

Al momento de enviar los flujos desde el controlador hacia los dispositivos Mikrotik se dio una prioridad de 10, es importante mencionar que el controlador OpenDayLight y el protocolo OpenFlow considera de mayor importancia a los flujos con mayor valor es decir entre 10 y 5, el valor de mayor prioridad es 10.

Al priorizar de esta manera el tráfico en los dispositivos de red se obtiene una mejor calidad de servicio aplicando el método de Diferenciación de Servicios (DiffServ) ya que todos los flujos que pertenezcan a VoIP tendrán una mayor prioridad que otros por los cuales viajaran otro tipo de datos.

Prioridad de flujos en el controlador

En el menú de la interfaz de controlador al seleccionar la opción *Yang UI* se debe ingresar a las siguientes opciones:

Opendaylight-inventory / config / nodes / node {id} / table {id} / flow {id}

A continuación se debe ingresar campos como el id del nodo a configurar y la prioridad como se muestra a continuación:

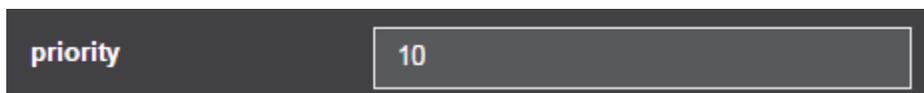
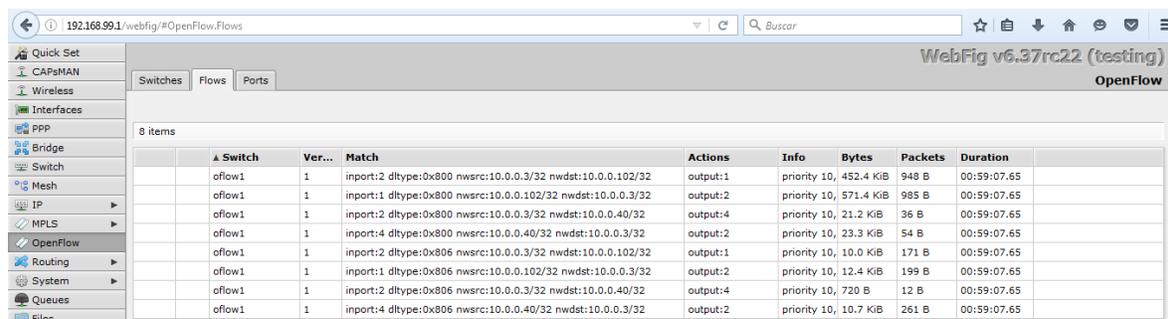


Imagen 62 Configuración de la prioridad

Flujos instalados en el dispositivo Mikrotik sdn1 (tabla OpenFlow)

Los flujos instalados en el dispositivo Mikrotik sdn1 son los siguientes:

A screenshot of a web browser showing the 'OpenFlow' configuration page in 'WebFig v6.37rc22 (testing)'. The 'Flows' tab is selected, displaying a table with 8 items. The table has columns for Switch, Ver..., Match, Actions, Info, Bytes, Packets, and Duration. The data shows various flow rules for switch 'oflow1' with different match criteria and actions.

	Switch	Ver...	Match	Actions	Info	Bytes	Packets	Duration
	oflow1	1	inport:2 dlttype:0x800 nwsrca:10.0.0.3/32 nwdst:10.0.0.102/32	output:1	priority 10,	452.4 KiB	948 B	00:59:07.65
	oflow1	1	inport:1 dlttype:0x800 nwsrca:10.0.0.102/32 nwdst:10.0.0.3/32	output:2	priority 10,	571.4 KiB	985 B	00:59:07.65
	oflow1	1	inport:2 dlttype:0x800 nwsrca:10.0.0.3/32 nwdst:10.0.0.40/32	output:4	priority 10,	21.2 KiB	36 B	00:59:07.65
	oflow1	1	inport:4 dlttype:0x800 nwsrca:10.0.0.40/32 nwdst:10.0.0.3/32	output:2	priority 10,	23.3 KiB	54 B	00:59:07.65
	oflow1	1	inport:2 dlttype:0x806 nwsrca:10.0.0.3/32 nwdst:10.0.0.102/32	output:1	priority 10,	10.0 KiB	171 B	00:59:07.65
	oflow1	1	inport:1 dlttype:0x806 nwsrca:10.0.0.102/32 nwdst:10.0.0.3/32	output:2	priority 10,	12.4 KiB	199 B	00:59:07.65
	oflow1	1	inport:2 dlttype:0x806 nwsrca:10.0.0.3/32 nwdst:10.0.0.40/32	output:4	priority 10,	720 B	12 B	00:59:07.65
	oflow1	1	inport:4 dlttype:0x806 nwsrca:10.0.0.40/32 nwdst:10.0.0.3/32	output:2	priority 10,	10.7 KiB	261 B	00:59:07.65

Imagen 63 Flujos instalados en el router 1

Flujos instalados en el dispositivo Mikrotik sdn2 (tabla OpenFlow)

Los flujos instalados en el dispositivo Mikrotik sdn2 son los siguientes:

Switch	Ver...	Match	Actions	Info	Bytes	Packets
oflow1	1	inport:1 dlttype:0x806 nwsrsc:10.0.0.3/32 nwdst:10.0.0.40/32	output:3	priority 10, idletimeou	720 B	12 B
oflow1	1	inport:1 dlttype:0x800 nwsrsc:10.0.0.3/32 nwdst:10.0.0.40/32	output:3	priority 10, idletimeou	20.0 KiB	34 B
oflow1	1	inport:3 dlttype:0x806 nwsrsc:10.0.0.40/32 nwdst:10.0.0.3/32	output:1	priority 10, idletimeou	4380 B	73 B
oflow1	1	inport:3 dlttype:0x800 nwsrsc:10.0.0.40/32 nwdst:10.0.0.3/32	output:1	priority 10, idletimeou	22.1 KiB	50 B
oflow1	1	inport:1 dlttype:0x806 nwsrsc:10.0.0.3/32 nwdst:10.0.0.20/32	output:5	priority 10, idletimeou	1140 B	19 B
oflow1	1	inport:1 dlttype:0x800 nwsrsc:10.0.0.3/32 nwdst:10.0.0.20/32	output:5	priority 10, idletimeou	147.9 KiB	622 B
oflow1	1	inport:5 dlttype:0x806 nwsrsc:10.0.0.20/32 nwdst:10.0.0.3/32	output:1	priority 10, idletimeou	798 B	19 B
oflow1	1	inport:5 dlttype:0x800 nwsrsc:10.0.0.20/32 nwdst:10.0.0.3/32	output:1	priority 10, idletimeou	155.4 KiB	662 B
oflow1	1	inport:3 dlttype:0x806 nwsrsc:10.0.0.102/32 nwdst:10.0.0.3/32	output:1	priority 10, idletimeou	10.6 KiB	169 B
oflow1	1	inport:4 dlttype:0x800	output:1	priority 10, idletimeou	470.9 KiB	4292 B
oflow1	1	inport:1 dlttype:0x800 nwsrsc:10.0.0.3/32 nwdst:10.0.0.200/32	output:2	priority 10, idletimeou	556.1 KiB	1459 B
oflow1	1	inport:2 dlttype:0x800 nwsrsc:10.0.0.200/32 nwdst:10.0.0.3/32	output:1	priority 10, idletimeou	682.5 KiB	1670 B
oflow1	1	inport:1 dlttype:0x800 nwsrsc:10.0.0.3/32 nwdst:10.0.0.102/32	output:3	priority 10, idletimeou	447.9 KiB	940 B
oflow1	1	inport:3 dlttype:0x800 nwsrsc:10.0.0.102/32 nwdst:10.0.0.3/32	output:1	priority 10, idletimeou	565.6 KiB	977 B
oflow1	1	inport:1 dlttype:0x806 nwsrsc:10.0.0.3/32 nwdst:10.0.0.102/32	output:3	priority 10, idletimeou	9.9 KiB	169 B
oflow1	1	inport:1 dlttype:0x806 nwsrsc:10.0.0.3/32 nwdst:10.0.0.200/32	output:2	priority 10, idletimeou	9.8 KiB	167 B
oflow1	1	inport:1 dlttype:0x806	output:4	priority 10, idletimeou	34.9 KiB	596 B
oflow1	1	inport:2 dlttype:0x806 nwsrsc:10.0.0.200/32 nwdst:10.0.0.3/32	output:1	priority 10, idletimeou	10.4 KiB	167 B
oflow1	1	inport:4 dlttype:0x806	output:1	priority 10, idletimeou	5.8 KiB	99 B
oflow1	1	inport:1 dlttype:0x800	output:4	priority 10, idletimeou	2250.8 KiB	11.8 KiB

Imagen 64 Flujos instalados en el router 2

Comparaciones de QoS

En los dos siguientes gráficos se puede observar los mismos flujos pero con distinta prioridad, en el caso de la primera imagen al tener una prioridad menor que otros flujos con prioridad 10, el tráfico es nulo y se mantiene a la espera de que se deje de transmitir datos en las interfaces para proceder a usar dichos flujos.

En la segunda imagen se configuro la prioridad con un valor de 15, por ende el tráfico empieza funcionar a través de estos flujos inmediatamente.

Trafico de flujos con menor prioridad (5)

Switch	Ver...	Match	Actions	Info	Bytes	Packets
oflow1	1	inport:1 dlttype:0x806 nwsrsc:10.0.0.3/32 nwdst:10.0.0.200/32	output:2	priority 5, idletimeout 0,	0 B	0 B
oflow1	1	inport:1 dlttype:0x800 nwsrsc:10.0.0.3/32 nwdst:10.0.0.200/32	output:2	priority 5, idletimeout 0,	0 B	0 B
oflow1	1	inport:2 dlttype:0x806 nwsrsc:10.0.0.200/32 nwdst:10.0.0.3/32	output:1	priority 5, idletimeout 0,	0 B	0 B
oflow1	1	inport:2 dlttype:0x800 nwsrsc:10.0.0.200/32 nwdst:10.0.0.3/32	output:1	priority 5, idletimeout 0,	0 B	0 B

Imagen 65 Verificación de prioridades en los flujos instalados

Trafico de flujos con mayor prioridad (15)

	▲ Switch	Ver...	Match	Actions	Info	Bytes	Packets
	oflow1	1	inport:1 dlttype:0x806 nwsrsc:10.0.0.3/32 nwdst:10.0.0.200/32	output:2	priority 15, idletimeout 0	2640 B	44 B
	oflow1	1	inport:1 dlttype:0x800 nwsrsc:10.0.0.3/32 nwdst:10.0.0.200/32	output:2	priority 15, idletimeout 0	7.7 MiB	37.0 KiB
	oflow1	1	inport:2 dlttype:0x806 nwsrsc:10.0.0.200/32 nwdst:10.0.0.3/32	output:1	priority 15, idletimeout 0	3776 B	59 B
	oflow1	1	inport:2 dlttype:0x800 nwsrsc:10.0.0.200/32 nwdst:10.0.0.3/32	output:1	priority 15, idletimeout 0	7.9 MiB	37.9 KiB

Imagen 66 Verificación de prioridades en los flujos instalados

CONCLUSIONES

Las redes SDN simplifican y flexibilizan la administración y despliegue de la red, gestionando de manera centralizada las tablas de flujo de datos, optimizando el tiempo de trabajo y reduciendo costos.

La recuperación de SDN ante fallos es ágil, versátil al aplicar alta disponibilidad y redundancia en el controlador optimiza la recuperación de desastres y se vuelve una alternativa a considerar en DRP así como en la virtualización del centro de datos SDDC (Centro de datos definido por software).

NaaS con neutron es una alternativa de puerta de enlace en la convergencia de redes tradicionales redes SDN y servicios en la nube, en nuestras pruebas se realizaron video llamadas en las que su origen inicial era la red tradicional y su destino era la red SDN y viceversa, siendo los resultados positivos en calidad y consumo de ancho de banda.

Muchas bibliografías manifiestan que SDN reemplaza a MPLS, en nuestra investigación concluimos que estas dos tecnologías puede convivir y empoderar las funcionalidades de la red combinando sus beneficios en el transporte de información, la ingeniería de tráfico y la división de los planos de control e infraestructura.

La aplicación de QoS en SDN se realiza mediante la clasificación de flujos, se establece una prioridad y se aplica algoritmo Diffserv, los flujos que no tienen prioridad son encolados a la espera de la liberación de recursos de la red, estas colas de flujos son tratadas sin prioridad sin embargo son enviadas a su destino por las tablas de reenvío ralentizando el arribo a la interfaz destino sin perder la información.

Aplicar políticas de QoS en toda la infraestructura, es simple basta con cambiar en el controlador los parámetros de los flujos y aplicar la calidad de servicio requerida por el servicio o cliente.

RECOMENDACIONES

Debido a que SDN es una manera eficaz de administrar las distintas infraestructuras de red no solo privadas, sino también a nivel de internet, es recomendable realizar más proyectos e investigaciones que aborden este tema ya que existen distintos controladores que ofrecen una gran variedad de funcionalidades con el protocolo OpenFlow, inclusive OpenDayLight tiene varios módulos que pueden configurar de distintas maneras la red de datos, como se observa en este trabajo técnico se configuraron varios dispositivos con diferentes flujos aplicando protocolos como IP, MPLS y políticas de Calidad de Servicio sin embargo este controlador Open Source ofrecen funcionalidades que aún deben ser investigadas para su aplicación en futuros proyectos.

Bibliografía

- [1] A. Sánchez Monge y K. Grzegorz Szarkowicz, MPLS en la época de la SDN, Juniper .
- [2] M. R. Ahamed Rahimi, H. Hashim y R. Ab Rahman, «Implementation of QoS in MPLS Networks,» p. 7, 2009.
- [3] M. Porwal, A. Yaday y S. Charchate, «Traffic Analysis of MPLS and Non MPLS Network including MPLS Signaling Protocols and Traffic distribution in OSPF and MPLS,» *IEEE*, p. 7.
- [4] M. Moro Vallina, Infraestructuras de redes de datos y sistemas de telefonía, Madrid: Paraninfo, 2013.
- [5] J. Barceló Ordinas, S. Llorente Viejo y X. Perramon Tornil, Protocolos y aplicaciones Internet, Barcelona: UOC, 2008.
- [6] E. Crawley, R. Nair, B. Rajagopalan y H. Sandick, *A Framework for QoS-based Routing in the Internet (IETF RFC 2386)*, IETF, 1998.
- [7] D. Zhang y D. Ionescu, «QoS Performance Analysis in Deployment of DiffServ-aware MPLS Traffic Engineering,» *IEEE*, p. 6.
- [8] K. Wallace, Implementing Cisco Unified Communications Voice over IP and QoS (CVOICE), Indianapolis: CISCOPRESS, 2011.
- [9] Zabbix, «The Ultimate Enterprise-class Monitoring Platform,» [En línea]. Available: <http://www.zabbix.com/>. [Último acceso: 10 09 2016].
- [10] V. Martínez de la Cruz, «Victoria Martínez de la Cruz,» 12 Julio 2013. [En línea]. Available: <http://vmartinezdelacruz.com/en-pocas-palabras-como-funciona-openstack/>.
- [11] T. Nolle, «TechTarget,» Octubre 2014. [En línea]. Available: <http://searchdatacenter.techtarget.com/es/cronica/Red-como-servicio-El-nucleo-para-la-conectividad-de-nube>.
- [12] M. Rouse , «SearchDataCenter en Español,» Enero 2014. [En línea]. Available: <http://searchdatacenter.techtarget.com/es/definicion/SDN-hibrida>. [Último acceso: 2016].
- [13] mininet.org, «Mininet Overview,» 2016. [En línea]. Available: <http://mininet.org/overview/>. [Último acceso: 09 10 2016].
- [14] R. Durner , A. Blenk y W. Kellerer, «Performance Study of Dynamic QoS Management for OpenFlow-enabled SDN Switches,» 2015.
- [15] M. Santos , B. Nunes, K. Obraczka y T. Turetli, «Decentralizing SDN's Control Plane,» *39th Annual IEEE Conference on Local Computer Networks*, 2014.

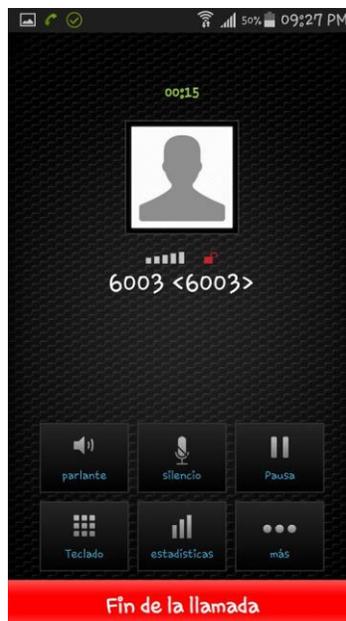
- [16] A. Gelberger, N. Yemini y R. Giladi, «Performance Analysis of Software-Defined Networking (SDN),» *2013 IEEE 21st International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems*, 2013.
- [17] S. Baik, Y. Lim, J. Kim y Y. Lee, «Adaptive flow monitoring in SDN architecture,» 2015.
- [18] H. Hata, «A Study of Requirements for SDN Switch Platform,» 2013.
- [19] C.-Y. Chu , K. Xi, M. Luo y J. Chao, «Congestion-Aware Single Link Failure Recovery in Hybrid SDN Networks,» 2015.
- [20] S. Baik, C. Hwang y Y. Lee, «SDN-based architecture for end-to-end path provisioning in the mixed circuit and packet network environment,» 2014.
- [21] P. H. Isolani, J. A. Wickboldt y C. B. Both, «Interactive Monitoring, Visualization, and Configuration of OpenFlow-Based SDN,» 2015.
- [22] O. Tkachova y M. J. Salim, «An Analysis of SDN-OpenStack Integration,» 2015.
- [23] C. Caba y J. Soler, «APIs for QoS configuration in Software Defined Networks,» *IEEE*, p. 5, 2015.
- [24] S. Dwarakanathan, L. Bass y L. Zhu, «Cloud Application HA using SDN to ensure QoS».
- [25] J. Kempf, S. White, J. Ellithorpe, P. Kazemian, M. Haitjema, N. Beheshti, S. Stuart y H. Green, «OpenFlow MPLS and the Open Source Label Switced Router,» p. 6.
- [26] L. Higgins, K. Hsu, G. McDowell, G. Nakamoto y W. Sax, «The QoS even manager - automated implementation of QoS Policies,» *IEEE*, p. 5.
- [27] K. Govindarajan, K. Chee Meng, H. Ong, W. Ming Tat, S. Sivanand y L. Swee Leong, «Realizing the QoS in SDN Based Cloud Infrastructure,» p. 7, 2014.
- [28] G. Ahn y W. Chun, «Design and Implementation of MPLS Network Simulator Supporting LDP and CR-LDP,» *IEEE*, p. 6.
- [29] T. Lee, «MPLS and DS as QoS Solutions,» *IEEE*, p. 5.
- [30] Citrix, «Introducción a Software Defined Networking,»
https://www.citrix.com/content/dam/citrix/en_us/documents/oth/sdn-101-an-introduction-to-software-defined-networking-es.pdf.

ANEXOS

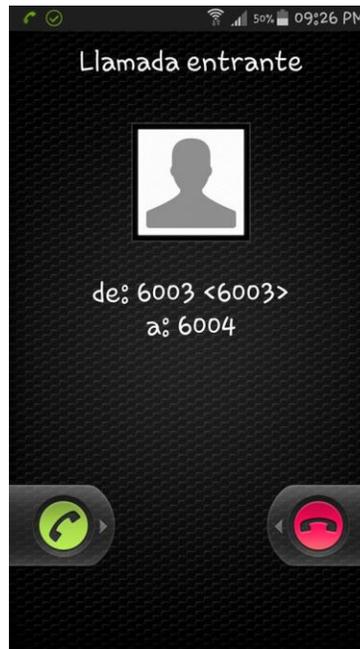
ANEXO 1. Ejemplo de código a enviar hacia los dispositivos de la red desde el controlador SDN.

```
Preview:
http://localhost:8181/restconf/config/opendaylight-inventory:nodes/node//flow-node-
inventory:table/0/flow/1
{
  "flow": [
    {
      "id": "1",
      "match": {
        "in-port": "1",
        "ethernet-match": {
          "ethernet-type": {
            "type": "2048"
          }
        }
      },
      "ipv4-source": "10.0.0.3/32",
      "ipv4-destination": "10.0.0.200/32"
    },
    {
      "instructions": {
        "instruction": [
          {
            "order": "0",
            "apply-actions": {
              "action": [
                {
                  "order": "0",
                  "output-action": {
                    "output-node-connector": "2",
                    "max-length": "100"
                  }
                }
              ]
            }
          }
        ]
      }
    }
  ]
}
```

ANEXO 2. Prueba de envío de llamada dentro de la red SDN.



ANEXO 3. Recepción de la llamada realizada dentro de la red SDN.



ANEXO 4. Establecimiento de la llamada entre los usuarios con las extensiones 6003 y 6004.

