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:

"Diseño y Despliegue de Funciones de Red Virtualizadas (NFV) usando Redes Definidas por Software (SDN) dentro de una infraestructura Virtual, aplicando balanceo de carga y seguridad distribuida en IPv6"

AUTOR:

Cordero Vizhñay Christian Fabián

TUTOR:

Ing. Pablo Leonidas Gallegos Segovia

Cuenca – Ecuador

2017

CESIÓN DE DERECHOS DE AUTOR

Yo, Christian Fabián Cordero Vizhñay con cedula de identidad N° 0104701990, manifiesto mi voluntad y cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autor del trabajo de titulación: "DISEÑO Y DESPLIEGUE DE FUNCIONES DE RED VIRTUALIZADAS (NFV) USANDO REDES DEFINIDAS POR SOFTWARE (SDN) DENTRO DE UNA INFRAESTRUCTURA VIRTUAL, APLICANDO BALANCEO DE CARGA Y SEGURIDAD DISTRIBUIDA EN IPV6", 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 mi condición de autor me reservo los derechos morales de la obra antes citada. En concordancia, suscribo este documento en el momento que hago entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Politécnica Salesiana.

Cuenca, octubre del 2017

_____

Christian Fabián Cordero Vizhñay

0104701990

CERTIFICACIÓN

Yo declaro que bajo mi tutoría, fue desarrollado el trabajo de titulación: “DISEÑO Y DESPLIEGUE DE FUNCIONES DE RED VIRTUALIZADAS (NFV) USANDO REDES DEFINIDAS POR SOFTWARE (SDN) DENTRO DE UNA INFRAESTRUCTURA VIRTUAL, APLICANDO BALANCEO DE CARGA Y SEGURIDAD DISTRIBUIDA EN IPV6”, realizado por: Christian Fabián Cordero Vizhñay, obteniendo este Proyecto Técnico que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana.

Cuenca, octubre del 2017

Ing. Pablo Gallegos Segovia

DECLARATORIA DE RESPONSABILIDAD

Yo, Christian Fabián Cordero Vizhñay con cedula de identidad N° 0104701990, del proyecto **"**DISEÑO Y DESPLIEGUE DE FUNCIONES DE RED VIRTUALIZADAS (NFV) USANDO REDES DEFINIDAS POR SOFTWARE (SDN) DENTRO DE UNA INFRAESTRUCTURA VIRTUAL, APLICANDO BALANCEO DE CARGA Y SEGURIDAD DISTRIBUIDA EN IPV6", certifico que el total contenido de este Proyecto Técnico es de mi exclusiva responsabilidad y autoría.

Cuenca, octubre del 2017

Christian Fabián Cordero Vizhñay

0104701990

AGRADECIMIENTO

Agradezco primeramente a Dios, a la Virgen María por la vida, por cuidarme y por darme las fuerzas de permitirme terminar este objetivo, a mi mami Patricia por ser mi guía y el apoyo fundamental para culminar esta meta, a mis abuelitos por haberme inculcado valores como el respeto, humildad, esfuerzo, perseverancia en seguir siempre adelante, a mi enamorada Natalí por su apoyo incondicional y siempre darme ánimos.

De igual modo a mi director de titulación por la guía, ayuda, paciencia y atención que me brindo para culminar el proyecto; finalmente a mis amigos y compañeros que me dieron sus consejos y el apoyo de manera desinteresada.

Christian Cordero

DEDICATORIA

Dedico este proyecto de titulación de manera especial a Dios, a la Virgen María, a mi mami, abuelitos, tíos, tías y a mi enamorada. A mi mami y mis abuelitos quienes fueron el pilar importante en mi vida universitaria, la razón para que cada día seguir adelante y ser una mejor persona. A mi enamorada que ha estado a mi lado en los momentos que tal vez deseaba rendirme estaba ahí para brindarme su fuerza y apoyo para cumplir esta meta en mi vida.

Este nuevo objetivo obtenido es gracias a ustedes, me han enseñado y aprendido varios valores importantes para la formación profesional y así terminar una meta más que al inicio me parecía imposible.

Christian Cordero

Índice

Índice de Figuras

Resumen

En este trabajo se muestra el diseño e implementación de las tecnologías de Redes Definidas por Software (SDN) y Funciones de Red Virtualizadas (NFV) en un entorno virtual, estas redes son programables lo cual nos permiten realizar una configuración y gestión de red sencilla, implementando la transición del protocolo IPv4 a Ipv6.

La implementación de SDN y NFV permitirán tener un mejor despliegue de aplicaciones, servicios y de infraestructura, con una flexibilidad y agilidad en menor tiempo, con la innovación de las funciones de red virtualizadas podremos tener la sustitución de nodos de red físicos en nodos virtuales.

En el proyecto se plantea un laboratorio de pruebas, implementando una arquitectura SDN que se va desplegar mediante el controlador de Opendaylight (ODL), en el cual se centraliza la administración y la gestión de dispositivos virtuales. Además se trabaja con un simulador como es mininet, para la virtualización de una red completa con hosts, switches y enlaces virtuales.

Con respecto a NFV hemos desarrollado en la plataforma Openstack (cloud privado), donde se crean las redes, routers virtuales y a su vez se despliegan las instancias (máquinas virtuales), aplicando seguridad y balanceo de carga.

Abstract

This article talk about the design and implementation of Software Defined Networks (SDN) and Virtual Network Functions (NFV) technologies in a virtual environment, these networks are programmable and allows us to perform a simple configuration and management of network, implementing the transition from IPv4 to IPv6.

The implementation of SDN and NFV will allow a better deployment of applications, services and infrastructure, with flexibility and agility in less time, with the innovation of virtualized network functions we can have the replacement of physical network nodes in virtual nodes.

In the project implementing an SDN architecture to be deployed by the Opendaylight (ODL) driver, which centralizes the management and virtual devices. In addition it works with an emulator like mininet, for the virtualization of a complete network with hosts, switches and virtual links.

With respect to NFV in the platform Openstack (private cloud) the networks are created, virtual routers and in turn are deployed the instances (virtual machines), applying security and load balancing.

Capítulo 1

Introducción

Motivación

La gestión de las tecnologías de la información (TI) requiere que las redes subyacentes transporten grandes cantidades de tráfico, permitan el despliegue de aplicaciones y servicios distintos y dinámicos. Así mismo la adopción de los conceptos de centros de datos definidos por software y la virtualización ha incrementado enormemente la demanda de la red. Además de diversidad y heterogeneidad del hardware de red desplegado, los protocolos distribuidos y componentes de software obligan a las redes heredadas a inundar con dispositivos de conmutación que deciden sobre la ruta tomada por cada paquete individualmente; además, las rutas de datos y los procesos de toma de decisiones para conmutación o enrutamiento se colocan en el mismo dispositivo. La capacidad de toma de decisiones o inteligencia de red se distribuye a través de los diversos componentes de hardware de red. Esto hace que la introducción de cualquier dispositivo o servicio de red nuevo sea un que costoso y que toma mucho tiempo porque requiere la reconfiguración de cada uno de los numerosos nodos de la red.

Los protocolos y estándares de las redes tradicionales fueron establecidos en bajo otros escenarios, y han sido muy efectivos hasta la actualidad, sin embargo, las redes heredadas no han evolucionado al mismo ritmo que las nuevas tecnologías de la información y comunicación, debido a que son estáticas, lentas para el cambio y dedicada a tareas específicas. Por lo que es necesario un nuevo salto tecnológico que revolucione a los entornos actuales y futuros. En especial si nos enfrentamos a nuevos retos como la migración de IPv4 a Ipv6, el internet de las cosas, el cloud computing, el Big Data, las ciudades inteligentes, las redes celulares de quinta generación (5G).

Es así que, para cubrir estas necesidades de entornos dinámicos, con interfaces programables que reducen la heterogeneidad de la red, con una administración y gestión centralizada nace el concepto de las redes definidas por software (SDN) y las Virtualización de las funciones de red (NFV).

El objetivo general de este proyecto es el de generar escenarios funcionales para validar o descartar el uso de las redes definidas por software y la Virtualización de las funciones de red como mecanismo de apoyo a la transición de los protocolos IPv4 ha IPv6.

En lo que continua de este documento realizaremos un análisis del estado del arte del proceso de migración de IPv4 hacia IPv6, continuaremos con la explicación del funcionamiento de la una red SDN y de las NFV, además presentaremos nuestros

escenario de evaluación y finalizaremos presentando nuestro aporte en esta investigación así como las conclusiones.

Alcance

El alcance de este proyecto, es probar y desplegar escenarios de simulación de red haciendo uso de NFV y SDN, para comprobar e despliegue de los mecanismos de transición sobre redes SDN. Se planteara diferentes entornos de haciendo uso del controlador de redes SDN Opendaylight en una configuración de cluster para alta disponibilidad y del emulador de redes Mininet en el cual se desplegaran scripts para la creación de topologías de red SDN. Finalmente mediante la utilización de Openstack se podrá implementar dispositivos de red virtuales e instancias que estarán funcionando en una nube privada, logrando de esta manera con la conjunción de todas estas tecnológicas una red dinámica, segura y flexible.

Delimitación

En este proyecto técnico de titulación se utilizara software de código abierto que realice las tareas de emulación de arquitecturas desplegadas que permitan evaluar el funcionamiento de SDN y NFV en escenarios de transición de los protocolos IP. Para ello usaremos el protocolo estándar de SDN que es Openflow, para la función de los dispositivos de reenvío de flujos SDN se utilizara el software llamado Mininet, para la gestión de la red SDN se utilizara el controlador Opendayligth y la nube privada que será desplegada en Openstack en la que se aplicaran los procesos de NFV.

Con el controlador SDN OpenDayLight asociado con mininet, se crearan mediante scripts desarrollados en Python diferentes escenarios de prueba; y mediante Openstack se implementaran dispositivos de red virtuales, se crearan reglas de Firewall para la seguridad y se desplegarán instancias (máquinas virtuales en la nube).

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

Objetivos

Objetivos Generales

Diseñar y desplegar un laboratorio de pruebas, empleando NFV y SDN. Generando una arquitectura que evidencie la “migración” de los protocolos IPV4 a IPV6.

Objetivos Específicos

- Establecer el estado del arte de NFV y SDN, y sus aplicaciones en redes de telecomunicaciones; y transición de IPv4 a IPv6.
- Desarrollar una arquitectura en la que se evaluara las funcionalidades de transición de Redes Definidas por Software y Virtualización de las Funciones de Red.
- Desarrollar un protocolo de pruebas y el análisis de resultados aplicando en el escenario.

Capítulo 2

Estado del arte

La internet es un red interconectada a nivel global, cuya autoridad de administración es la Autoridad de asignación de número IANA (por sus siglas en inglés) proporciona dos versiones actuales del Protocolo de Internet (IP). Internet Protocol versión 4, definido en la RFC 791 y IPv6 (Internet Protocol versión 6, definido en la RFC 2460).

IPv4 se basa en un pool de direcciones codificadas en 32 bits, por tanto, son un aproximado de 4.294.967.296. Existen tres tipos de direcciones IPv4.

Direcciones IPv4 públicas

Las direcciones IPv4 públicas constituyen el espacio de direcciones de Internet. Estas son distribuidas para ser globalmente únicas de acuerdos a los objetivos que se describirán más adelante en este documento. El principal propósito de este espacio de direcciones es permitir la comunicación usando el IPv4 sobre Internet.

Un propósito secundario es permitir la comunicación entre redes privadas interconectadas[1].

Direcciones IPv4 privadas

Algunos rangos de direcciones IPv4 han sido reservados para la operación de redes privadas. Cualquier organización puede usar estas direcciones IPv4 en sus redes privadas sin la necesidad de solicitarlo a algún Registro de Internet. La principal condición establecida para el uso de direcciones IPv4 privadas es que los dispositivos que usen estas direcciones IPv4 no necesiten ser alcanzados desde Internet.

Para una descripción más detallada acerca del espacio de direcciones IPv4 privadas, por favor consulte el RFC 1918 [1]

Direcciones IPv4 especiales y reservadas

Estas son rangos de direcciones IPv4 reservadas para aplicaciones como el multicasting, estas direcciones IPv4 están descritas en el RFC 1112 y para propósitos de este capítulo están más allá del contexto del mismo.

El Sistema de registro de Internet

El sistema de registro de Internet ha sido establecido con la finalidad de hacer cumplir los objetivos de exclusividad, conservación, ruteabilidad e información. Este sistema consiste de Registros de Internet (IR) organizados jerárquicamente. Los espacios de direcciones IPv4 son típicamente asignados a los usuarios finales por los ISPs o los

[1] http://www.lacnic.net/545/1/lacnic/2-direcciones-ipv4

NIRs.

Por otra parte, estos espacios de direcciones IPv4 son previamente distribuidos a los NIRs e ISPs por parte de los Registros Regionales de Internet.[1]

Bajo este sistema los usuarios finales son aquellas organizaciones que operan redes en donde se utilizan los espacios de direcciones IPv4. Los NIRs al igual que LACNIC mantienen espacios de direcciones IPv4 para ser asignados a usuarios finales o distribuidos a Proveedores de Servicios de Internet. Los espacios de direcciones IPv4 asignados son utilizados para la operación de redes, mientras que el espacio de direcciones IPv4 distribuidos se mantiene en los Registros de Internet para futuras asignaciones a sus usuarios finales[1].

Esta distribución ha sido funcional hasta el año 2000 en que se inició una explosión del uso del internet como se ve en la grafico [1] En el cual su crecimiento se aceleró, los dispositivos inteligentes y su integración con el internet generaron una gran demanda de direcciones públicas las mismas que no fueron pensada en los años 70.

Es así que para Vinton 'Vint' G. Cerf y Robert Elliot Kahn, los considerados padres de Internet. Vint Cert, que es el Chief Internet Evangelist de Google y presidente de la ICANN, recientemente, en una entrevista para el Sidney Morning Herald comentó que:

"Pensé que era un experimento y que 4.300 millones direcciones serían suficientes. ¿Quién diablos sabía cuántas direcciones se necesitaban?" Esto no significa que Internet se pare, simplemente significa que no se ha construido muy bien. [2]

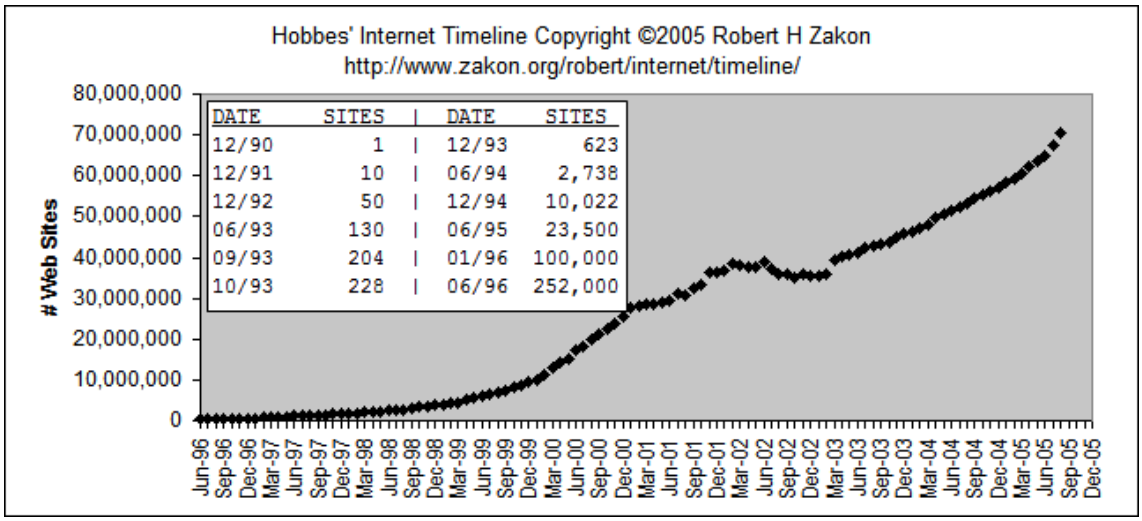

Figura 1 Gráfica de crecimiento del internet de 1996 – 2005 [1]

[2] https://hipertextual.com/archivo/2011/02/transicion-ipv4-a-ipv6-lo-que-necesitas-saber/

El pasado 3 de Febrero de 2011 se ha entregado las últimas direcciones IPv4 del registro central (IANA). En el caso de Europa (RIPE NCC) y en Latinoamérica LANIC, éstas se agotarán previsiblemente a final de 2011, y a continuación, los ISPs agotarán sus propios recursos, también en pocos meses. Es por eso que se inició el proceso de transición desde la actual versión IPv4 hacia la nueva versión IPv6, pese a que en algunos países no existen direcciones disponibles en Ecuador algunos operadores como es el caso de Etapa Ep, cuenta un número aproximado de 65 mil direcciones IP públicas disponibles.[3]

¿Pero qué es la Transición a IPv6?

Como vimos en los párrafos anteriores el protocolo predominante en la actualidad en Internet es IPv4, e Internet se ha convertido en algo vital, no es posible su sustitución, es decir, no es posible apagar la Red, ni siquiera por unos minutos y cambiar a IPv6.

Tanto el protocolo IPv4 y el protocolo IPv6 no son compatibles, es decir los servicios que corren en el uno no funcionan en el otro por lo que basta con actualizar unos pocos equipos, es una operación que tendría que involucrar a cualquier organización, sea empresa, administración pública o proveedor de acceso o contenidos de una forma sincronizada, lo cual es imposible.

Precisamente por ello, la organización encargada de la estandarización de los protocolos de Internet (IETF, Internet Engineering Task Force), diseñó junto con el propio IPv6, una serie de mecanismos que llamamos de transición y coexistencia[4].

Básicamente es importante entender lo que ello implica. No se trata de una migración como erróneamente se indica en muchas ocasiones, sino que ambos protocolos, IPv4 e IPv6, existirán durante algún tiempo, es decir se produce una coexistencia[4].

El nuevo protocolo IPv6, permite direccionar alrededor de trescientos cuarenta sextillones de dispositivos, que frente a IPv4 es incomparable. El nuevo protocolo está siendo ya implementado en las redes y lo que se espera, es que coexista con IPv4 hasta que todo sea manejado con IPv6. Entonces, se necesita de un mecanismo de transición que permita manejar los dos protocolos, en tanto que IPv4 siga existiendo.

[3] https://www.telecomunicaciones.gob.ec/se-agotan-dominios-ipv4-pero-en-ecuador-se-fortalece-protocolo-ipv6/

[4] http://www.ipv6.es/es-ES/transicion/quees/Paginas/Transicion.aspx

En Ecuador, el Ministerio de Telecomunicaciones y Sociedad de la información, mediante acuerdo ministerial No. 007-2012, ejecutó un plan de acciones para que se lleve a cabo una transición ordenada y coexistente de IPv4 a IPv6 con operadores, ISP (Proveedor de Internet), entidades y organismos del sector público y privado. Es por ello, que en sus plataformas electrónicas debe empezar a generar tráfico IPv6. [5]

En el siguiente cuadro realizamos la comparativa entre las dos versiones del protocolo, para esclarecer el por qué el nuevo protocolo IP ofrece más ventajas que su versión actual.

Comparación entre IPv4 e IPv6	
IPv4	IPv6
Las direcciones tanto de origen como destino son de 32 bits de longitud (4 Bytes)	Las direcciones de origen y destino son de 128 bits de longitud (16 Bytes).
IPSec es un protocolo opcional.	IPSec es nativo.
No existe identificación de paquetes QoS	El campo flow label maneja el QoS
La fragmentación se la realiza a lo largo de la ruta de los encaminadores	La fragmentación se la realiza en el router origen y los router intermedio no realizan fragmentación
Cuenta con Broadcast que ralentiza la red	Elimina el Broadcast

Tabla 1 Comparación del protocolo IPv4 e IPv6

De estas características descritas se desprende los motivos que nos llevan hacia esta transición: El principal, el agotamiento de las direcciones IP, además de los nuevas paradigmas que requieren de mayor seguridad como es el caso del internet de las cosas (IoT), el comercio electrónico, las plataformas en la nube, la computación al borde, el Big data y todos los objetos inteligentes como celulares, tabletas , gatgets que requieren acceso a la Internet e identidad dentro de la red ya que generan gran cantidad de información, que requiere ser transportada y procesada con mecanismos que garantice su confidencialidad integridad y disponibilidad.

[5] https://www.telecomunicaciones.gob.ec/wp-content/uploads/downloads/2012/07/02_14_Acuerdo_No_007-2012.pdf

Mecanismos de transición

Debido a la incompatibilidad de IPv4 e IPv6, se desarrollaron varios métodos para la transición a IPv6. Sin embargo, el despliegue de IPv6 y los mecanismos de transición en las redes de producción han sido bajos y han aumentado lentamente. Cada mecanismo tiene detalles que lo hacen adecuado para un uso particular, por lo que la combinación de varios métodos se usa en la práctica.

Una pila dual (dual stack) es el mecanismo de transición más fácil de implementar: cada dispositivo implementa las pilas de protocolos IPv4 e IPv6 que se usan en paralelo. Por otro lado, este enfoque desperdicia recursos disponibles y requiere soporte para dos redes lógicas [2].

Además de una pila dual, hay 2 categorías principales de métodos de transición, [4]:

Tunelización. (Tunneling en Ingles)

Los mecanismos de transición de túnel usan encapsulación para el tránsito de datagramas IPv6 en paquetes IPv4 usando el protocolo número 41. El túnel se usa para conectar islas de red IPv6 a través de la red IPv4. Si bien el proceso de túneles en sí es el mismo para todos los mecanismos de transición de túneles, la configuración del túnel puede ser automática o manual [3]. Los túneles creados manualmente pueden ser entre enrutadores o puntos finales, según la configuración. IPv6 tunnel broker se puede usar para ayudar en la creación de túneles en combinación con TSP (Tunnel Setup Protocol) o TIC (Tunnel Information Protocol).

La técnica de tunelización más simple con configuración manual y la menor sobrecarga es 6in4 [5] que usa enrutadores de doble pila en los límites de la red que encapsulan datagramas IPv6 con encabezado IPv4 20B de longitud fija. 6to4 es un método automático basado en 6in4 con enrutadores que usan un prefijo IPv6 especial que consta de 2002 :: / 16 y su dirección IPv4 pública. El prefijo resultante / 48 permite más subredes y aumenta la automatización de la configuración del enrutador. Sin embargo, si un enrutador delimitador está detrás de NAT y no tiene una dirección IPv4 pública, debe usar la dirección IPv4 pública del enrutador ascendente. La implementación rápida de IPv6 (6ta) es un mecanismo basado en 6to4. Resuelve las fallas 6to4 mediante el uso del prefijo IPv6 propio de los ISP y el mapeo de direcciones IPv4 en lugar de 2002 :: / 16prefijo utilizado en 6to4 para garantizar la accesibilidad de la red IPv6 desde fuera de la red ISP [6].

El mecanismo de transición 6over4 se centra en hosts IPv6 aislados con una doble pila en redes habilitadas para multidifusión con hosts predominantemente IPv4. ISATAP es un mecanismo de transición más complicado análogo a 6over4, pero sin la necesidad

de soporte de multidifusión en la red IPv4.

Teredo es un mecanismo automático para hosts IPv6 aislados que pueden atravesar NAT. Lo hace encapsulando el tráfico IPv6 en UDPv4, pero este proceso hace que Teredo sea poco fiable e ineficiente [6].

Los mecanismos de transición de traducción [3] funcionan de forma similar a NAT en IPv4: los enrutadores de doble pila traducen las direcciones IPv4 e IPv6 en los límites de la red. Sin embargo, en lugar de cambiar solo los campos de dirección, los encabezados completos se cambian a IPv6 o IPv4. Este enfoque permite la comunicación de hosts en redes IPv6 con hosts en redes IPv4. Los problemas con el uso de NAT-Protocol Translation descritos en RFC4966 (principalmente la interacción con DNS) llevaron a su desaprobación. NAT64 utilizado principalmente para la comunicación iniciada por hosts en IPv6 es el único representante actual de los mecanismos de transición de traducción.

Cada uno de estos procesos requieren cambios en los encaminadores, la gran mayoría de los cambios se realizan de manera manual y lo que conlleva a multiplicar las posibilidades de fallo en la red, en especial cuando se trata de mecanismos de traducción y tuneles. Es por esta razón que es necesario realizar la automatización de los mecanismos de transición dentro de la red , sin embargo las redes tradicionales no cuentan con interfaces abiertas y están atadas a las especificaciones que brinda el fabricante lo que conlleva a una sistema complejo y heterogéneo, en este escenario los modelos de redes de nueva generación SDN y NFV son una alternativa que se debe analizar y validar para contar con esta herramientas como alternativas de migración o coexistencia.

Capítulo 3

Las redes definidas por software (SDN).

Definición

El embrión de SDN se creó por primera vez en 2006 en el proyecto Clean-State de Stanford. Ethane [8], un enfoque de control centralizado y programable en la red empresarial, presentó el método SDN en su etapa inicial. Como su nombre lo indica, SDN es una nueva arquitectura de red basada en lógica de software definida por el usuario, y el concepto de diseño de la arquitectura SDN es separar la lógica de control de los dispositivos de reenvío. Por lo tanto, el controlador centralizado tendrá una visión global de los recursos de la red, lo cual es esencial para la optimización de la red, por ejemplo, para mejorar la utilización del ancho de banda y garantizar transmisiones de red diferenciales para tráfico diferente. Además, SDN es más flexible que la red tradicional debido a su programabilidad, lo que permite a los usuarios desarrollar aplicaciones para controlar su red. Por lo tanto, los nuevos servicios podrían desplegarse rápidamente y los costos de administración y mantenimiento de la operación (OAM) se reducirán drásticamente. [9]

El objetivo de la red definida por software (SDN) es permitir que los ingenieros y administradores de la nube y la red respondan rápidamente a los cambios en los requisitos del negocio a través de una consola de control centralizada. SDN abarca múltiples tipos de tecnologías de red diseñadas para hacer que la red sea más flexible y ágil para admitir el servidor virtualizado y la infraestructura de almacenamiento del centro de datos moderno. Las redes definidas por software definieron originalmente un enfoque para diseñar, construir y gestionar redes que separa los planos de control (cerebros) y de reenvío (musculares) de la red, permitiendo así que el control de la red sea directamente programable y la infraestructura subyacente abstraída para aplicaciones y redes servicios.[6]

El plano de control de SDN dicta el comportamiento de la red completa. Esta centralización lógica introduce una forma más simple y más flexible de administrar y controlar el tráfico de red mediante un protocolo "hacia el sur" (OpenFlow). El protocolo OpenFlow permite que el plano de control instale reglas de flujo en el plano de datos. El plano de datos seguirá estas reglas de flujo para manejar los flujos de red. Cuando llega un nuevo flujo de falta de tabla, no coincide con ninguna de las reglas de flujo existentes. El plano de datos enviará un mensaje packet_in al plano de control para obtener instrucciones. [7] Mientras la capa de aplicación. Consiste en las aplicaciones de

[6] http://sdntutorials.com/difference-between-control-plane-and-data-plane/

negocio de los usuarios finales, que utilizan servicios de comunicación de SDN a través de las API hacia arriba (northbound) de la capa de control, tales como REST, JSON, XML, etc., permite a los servicios y aplicaciones simplificar y automatizar las tareas de configuración, provisión y gestionar nuevos servicios en la red, ofreciendo a los operadores nuevas vías de ingresos, diferenciación e innovación.[7]

Las diferencias éntrelos las funciones que cumplen los planos de control y de datos son:

Plano de control

Toma decisiones sobre dónde se envía el tráfico

Los paquetes del plano de control están destinados u originados localmente por el enrutador mismo

Las funciones del plano de control incluyen la configuración del sistema, la administración y el intercambio de la información de la tabla de enrutamiento

El controlador de ruta intercambia la información de topología con otros enrutadores y construye una tabla de enrutamiento basada en un protocolo de enrutamiento, por ejemplo, RIP, OSPF o BGP.

El enrutador procesa los paquetes del plano de control para actualizar la información de la tabla de enrutamiento.

Es la Señalización de la red

Como las funciones de control no se realizan en cada paquete individual que llega, no tienen una restricción de velocidad estricta y son menos críticas en cuanto al tiempo [8]

Plano de datos

También conocido como plano de reenvío

Reenvía el tráfico al siguiente salto a lo largo de la ruta a la red de destino seleccionada según la lógica del plano de control

Los paquetes de plano de datos pasan por el enrutador

Los enrutadores / conmutadores usan lo que el plano de control construyó para

[7] https://www.ramonmillan.com/tutoriales/sdnredesinteligentes.php

[8] https://learningnetwork.cisco.com/thread/33735

eliminar los marcos y paquetes entrantes y salientes

Con los siguientes ejemplos cuya fuente están citados al pie de pagina[6-7] queremos dejar claro el funcionamiento de los dos planes que rigen la red SDN.

Ejemplo 1

El protocolo o aplicación en sí mismo no determina realmente si el tráfico es control, administración o plano de datos, pero lo más importante es cómo lo procesa el enrutador. Considere una topología de 3 enrutadores con los enrutadores R1, R2 y R3. Digamos que se establece una sesión de Telnet desde R1 a R3. En ambos enrutadores, los paquetes deben ser manejados por el plano de control / gestión. Sin embargo, desde la perspectiva de R2, esto es solo tráfico de plano de datos que está en tránsito entre sus enlaces.

Ejemplo 2

Plano de control => Aprender lo que haremos

Nuestra etapa de planificación, que incluye aprender qué rutas tomarán los autobuses, es similar al plano de control en la red. Todavía no hemos recogido personas, ni las hemos dejado, pero conocemos los caminos y las paradas debido a nuestro plan. El plano de control se trata principalmente del aprendizaje de rutas.

Plano de datos => En realidad, mover los paquetes en función de lo que hemos aprendido.

El plano de datos es el movimiento real de los paquetes de datos de los clientes sobre la ruta de tránsito que aprendimos en la etapa del avión de control.

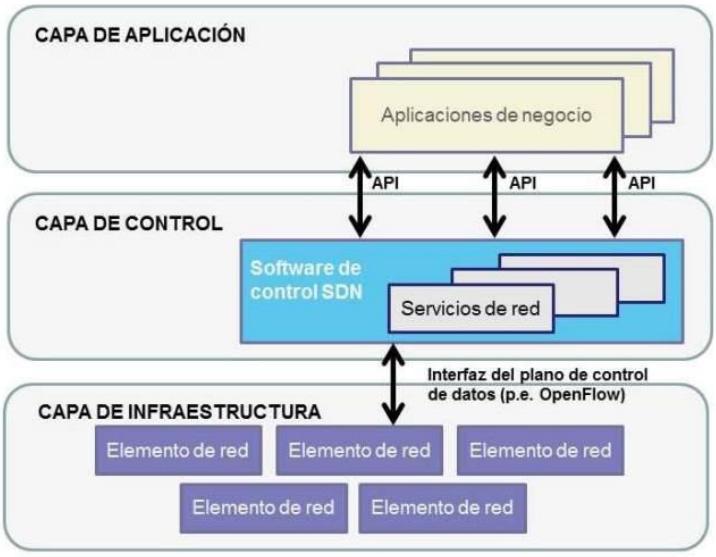

Figura 2 Arquitectura de SDN

¿Cómo funciona la red definida por software o SDN?

El control de la red al separar la lógica de control de los recursos de la computadora fuera del dispositivo. Todos los modelos SDN tienen alguna versión de un Controlador SDN, así como API hacia el sur y API hacia el norte:

Controladores: los "cerebros" de la red, los controladores SDN ofrecen una vista centralizada de la red general, y permiten a los administradores de red dictar a los sistemas subyacentes (como conmutadores y enrutadores) cómo debe controlar el tráfico de red el plano de reenvío.

API hacia el sur: las redes definidas por software usan API hacia el sur para transmitir información a los conmutadores y enrutadores "debajo". Openflow, considerado el primer estándar en SDN, era la API original en dirección sur y sigue siendo uno de los protocolos más comunes. A pesar de que algunos consideran que OpenFlow y SDN son uno en lo mismo, OpenFlow es solo una parte del panorama de SDN más grande.

API hacia el norte: las redes definidas por software usan API hacia el norte para comunicarse con las aplicaciones y la lógica de negocios "arriba". Estas ayudan a los administradores de red a configurar el tráfico y a implementar servicios de manera programática[9].

El protocolo Openflow

OpenFlow es la base (protocolo hacia el sur) de SDN. Open Networking Foundation es responsable de promover OpenFlow y SDN y actualmente hay más de cien miembros [11]. Así que OpenFlow se ve cada vez más en dispositivos de red. El origen de OpenFlow se remonta a Ethane (2006), que también se desarrolló en la Universidad de Stanford [2]. OpenFlow separa el plano de control del plano de datos de un conmutador o enrutador Ethernet común. Mientras el plano de datos permanece en el conmutador, el plano de control se ubica y gestiona desde una nueva entidad de red denominada Controlador OpenFlow o simplemente Controlador. El controlador tiene la capacidad de hablar con muchos conmutadores habilitados para OpenFlow en la red y administrar el tráfico de acuerdo con las políticas o las decisiones del administrador. Esto permite configuraciones de red más flexibles y eficientes cuando se combina con protocolos SDN hacia el norte[13].

[9] https://www.sdxcentral.com/sdn/definitions/what-the-definition-of-software-defined-networking-sdn/

El protocolo OpenFlow[6] consta de tres tipos diferentes de mensajes:

1. Mensaje del controlador al switch: Lo inicia el controlador, y tiene la función de conocer y actuar sobre el estado del switch. Le envía una petición de características al switch y este debe responder con las capacidades disponibles.

Podemos distinguir cuatro tipos dentro de este mensaje:

Modify-State: Su objetivo principal es añadir y eliminar entradas en las tablas de flujo y fijas características de los puertos.

Read-State: pregunta al switch por estadísticas del tráfico.

Send-Packet: envía paquetes por un puerto específico en el switch.

Barrier request/reply: se usa para recibir notificaciones de operaciones completadas.

2. Mensajes asíncronos: Existen cuatro tipos de mensajes que los switches envían al controlador tras la llegada de un paquete, tras un error, o tras un cambio de estado:

- Packet-in: es enviado al controlador cuando se recibe cualquier paquete que no tenga coincidencias en la tabla de flujo, o si la acción que corresponde en la tabla coincidente es reenviar el paquete al controlador.
- Flow-modify: enviado cuando una entrada es añadida a la entrada de flujos de un switch.
- Port-status: enviado al cambiar el estado de un puesto determinado.
- Error.

3. Mensajes simétricos: Se envían sin solicitud en cualquier dirección, y se distinguen tres tipos:

- Hello: mensajes intercambiados durante la negociación de una conexión.
- Echo request/reply: pueden ser enviados por el switch o por el controlador, con el fin de comprobar la comunicaci´on entre ellos. Debe ser respondido con un mensaje Echo replpy.
- Vendor message: está pensado para futuras revisiones de OpenFlow, proporciona una manera de ofrecer características adicionales.

Un switch OpenFlow puede tener diferentes tablas de flujos, en las cuales se realiza la búsqueda de coincidencia con los paquetes entrantes. Cada entrada en una tabla de flujos consta de tres campos principalmente:

1. Una cabecera que define el flujo.

2. Contadores para llevar la cuenta de la coincidencia de paquetes que pueden actualizarse por flujo, por tabla, puesto y por cola.

3. Acciones a realizar cuando se encuentra una coincidencia entre un paquete y una tabla de flujos.

Una entrada en la tabla de flujo se identifica por el conjunto de los campos Match fields y Priority, y tendrá un formato definido.

Match fields: contiene puerto y cabecera, y opcionalmente metadatos especificados en una tabla anterior.

Priority: determina la prioridad de la entrada de flujo.

Counters: se actualiza cuando se encuentran coincidencias.

Instructions: acciones que serán ejecutadas si los paquetes se relacionan con alguna entrada de la tabla de flujo.

Timeout: tiempo máximo de espera antes de que un flujo caduque.

Cookie: es un valor usado por el controlador para modificaciones del flujo, filtrar estadísticas, No es usado cuando se están procesando paquetes.

En una red con OpenFlow, sin embargo todas estas reglas pueden modificarse, quitar y añadir otras, según los objetivos concretos que se buscan. Por ejemplo, para reducir correspondiente al tráfico de correo electrónico para dar prioridad al streaming de video, etc.

En resumen. El protocolo OpenFlow indica al tráfico como fluir por la red; las reglas son definidas para cada flujo, con lo que se mantiene un control más exhaustivo de la red.

Nuestro análisis nos permite afirmar que SDN NO es Openflow: OpenFlow es sinónimo de redes definidas por software, pero es solo un elemento único en la arquitectura SDN general. OpenFlow es un estándar abierto para un protocolo de comunicaciones que permite que el plano de control interactúe con el plano de reenvío. Debe tenerse en cuenta que OpenFlow no es el único protocolo disponible o en desarrollo para SDN como es el caso de Cisco Open Network Environment (Cisco ONE), promete hacer que

toda la red sea programable, aunque se centra más en nuevas capas de red con APIs abiertas que en centralizar el control y gestión de la red con un controlador[9].

Mininet, este es otro de los componentes de sistemas de emulación el que usaremos con el propósito de utilizarlo como reenviador de flujos a través de la red a continuación realizaos una breve introducción a este componente:

Mininet

Mininet [14], es un programa emulador de red que posee una colección de hosts finales(los cuales se ejecutan en software Linux), enlaces de red, conmutadores virtuales y controladores en un solo núcleo de Linux, utilizado para realizar rápidamente experimentos con SDN y OpenFlow. Mediante una interfaz de línea de comandos (CLI) con la que se puede interactuar y verificar la red.

Ya que Mininet provee de un entorno simulado para la experimentación, se puede desarrollar y probar nuevas ideas mediante el desarrollo de scripts, antes de desplegarlas en un ambiente real. Además, el contorno de mininet está abierto a la corrección de fallos por parte de cualquier usuario y se distribuye bajo la licencia de BSD de código abierto. En el desarrollo de nuevas topologías de red, el acceso es permitido mediante OpenFlow mediante el puerto TCP 6634.

Características de Mininet

Mininet se caracteriza por:

- *Sencillez.-* Con un solo comando sudo mn se puede desplegar una red rápidamente.
- *Personalización.-* Se puede crear cualquier tipo de topología, mediante el lenguaje de programación Python, se puede realizar scripts para experimentar y almacenar nuevos ejemplos de topologías.
- *Entorno real.-* Se emula el envío de paquetes a través de interfaces reales Ethernet, con su velocidad de enlace y retardo. Se permite el diagnóstico con herramientas como ping, dump o tcpdump para comprobar la conectividad entre dos hosts

Comandos Principales

Para obtener la información sobre la topología que se ha desarrollado, es importante saber los siguientes comandos de mininet:

- Para visualizar los nodos que se crean en la red, se utiliza el comando **“nodes”**, como se observa en la Figura 3**:**

```
mininet> nodes
available nodes are:
c0 c1 h1 h10 h2 h3 h4 h5 h6 h7 h8 h9 s1 s2
```

Figura 3 Ejecución del comando “nodes”

- Los puertos para cada interfaz implementada y que se conectan con los dispositivos, se pueden observar en la Figura 4 al ejecutar **“ports”:**

```
mininet> ports
s1 lo:0 s1-eth1:1 s1-eth2:2 s1-eth3:3 s1-eth4:4 s1-eth5:5 s1-eth6:6
s2 lo:0 s2-eth1:1 s2-eth2:2 s2-eth3:3 s2-eth4:4 s2-eth5:5 s2-eth6:6
```

Figura 4 Ejecución del comando “ports”

- Los enlaces de la topología que se establecen, se puede visualizar en la Figura 5 escribiendo **“net”**:

```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
h3 h3-eth0:s1-eth3
h4 h4-eth0:s1-eth4
h5 h5-eth0:s1-eth5
h6 h6-eth0:s2-eth1
h7 h7-eth0:s2-eth2
h8 h8-eth0:s2-eth3
h9 h9-eth0:s2-eth4
h10 h10-eth0:s2-eth5
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0 s1-eth3:h3-eth0 s1-eth4:h4-eth0 s1-eth5:h5-eth0 s1-eth6:s2-eth6
s2 lo:  s2-eth1:h6-eth0 s2-eth2:h7-eth0 s2-eth3:h8-eth0 s2-eth4:h9-eth0 s2-eth5:h10-eth0 s2-eth6:s1-eth6
c0
c1
```

Figura 5 Ejecución del comando “net”

- Para observar las interfaces que tiene cada componente, se utiliza **“intfs”** como se muestra en la Figura 6:

```
mininet> intfs
h1: h1-eth0
h2: h2-eth0
h3: h3-eth0
h4: h4-eth0
h5: h5-eth0
h6: h6-eth0
h7: h7-eth0
h8: h8-eth0
h9: h9-eth0
h10: h10-eth0
s1: lo,s1-eth1,s1-eth2,s1-eth3,s1-eth4,s1-eth5,s1-eth6
s2: lo,s2-eth1,s2-eth2,s2-eth3,s2-eth4,s2-eth5,s2-eth6
c0:
c1:
```

Figura 6 Ejecución del comando "intfs"

- Para observar la información como el nombre de cada componente, las direcciones IP, las interfaces, los switches y los controladores, se digita el comando **"dump"** como se observa en la Figura 7:

```
mininet> dump
<Host h1: h1-eth0:10.1.1.1 pid=99531>
<Host h2: h2-eth0:10.1.1.2 pid=99535>
<Host h3: h3-eth0:10.1.1.3 pid=99537>
<Host h4: h4-eth0:10.1.1.4 pid=99539>
<Host h5: h5-eth0:10.1.1.5 pid=99541>
<Host h6: h6-eth0:10.1.1.6 pid=99543>
<Host h7: h7-eth0:10.1.1.7 pid=99545>
<Host h8: h8-eth0:10.1.1.8 pid=99547>
<Host h9: h9-eth0:10.1.1.9 pid=99549>
<Host h10: h10-eth0:10.1.1.10 pid=99551>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None,s1-eth4:None,s1-eth5:None,s1-eth6:None pid=99556>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None,s2-eth4:None,s2-eth5:None,s2-eth6:None pid=99559>
<RemoteController c0: 127.0.0.1:6633 pid=99525>
<RemoteController c1: 192.168.1.111:6633 pid=99725>
```

Figura 7 Ejecución del comando "dump"

- Para la comprobación de conectividad entre todos los hosts, para constancia de los nuevos elementos que contiene el controlador, se utiliza el comando **"pingall"** como se muestra en la Figura 8:

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Results: 0% dropped (90/90 received)
```

Figura 8 Prueba de conectividad mediante "pingall"

- Para salir de la red, digitamos el siguiente comando **"exit"**, como se visualiza en la Figura 9:

```
mininet> exit
*** Stopping network
```

Figura 9 Ejecución del comando "exit"

OpenDayLight

OpenDayLight[10] es una plataforma de Código Abierto (Open Source), que nos permite personalizar, acelerar, automatizar el diseño e implementación de Redes Definidas por Software (SDN) [15].

El proyecto OpenDaylight surgió de la tendencia innovadora de SDN, con un objetivo principal que es la programación de la red. Es un controlador de OpenFlow, su función

[10] https://www.opendaylight.org/what-we-do/current-release

es centralizar las decisiones de envío y recepción de paquetes de modo que la red se puede programar independiente de los conmutadores individuales y equipo del centro de datos (CD).

Como se define en [16]: *"OpenDayLight pretende convertirse en una plataforma abierta que utilicen todas las empresas, evitando que las aplicaciones (programas) privativas (propietarios) restrinjan el crecimiento del mercado y al mismo tiempo se reduzcan los costes de desarrollo".*

OpenDayLight incorpora áreas como: el controlador, protocolos de las aplicaciones de red, los switches virtuales, proporciona gran escalabilidad, flexibilidad y capacidad de programación de red de los switches y routers basados en software [15] [16].

OpenDayLight está desarrollado en karaf. Karaf es una tecnología de contenedores, que a los desarrolladores de software les permiten instalar módulos necesarios para la compilación y todo ello en un solo paquete de distribución [17].

Aspectos principales de la arquitectura de ODL
Los aspectos principales tenemos [18]:

API REST.- es la interfaz de comunicación hacia el norte, está diseñada para la integración con plataformas como OpenStack (neutrón) y aplicaciones personalizadas. La API REST se utiliza para crear una interfaz gráfica de usuario o GUI. El API REST de ODL además permite que las aplicaciones de redes específicas se construyan desde afuera de ODL, con el fin de aprovechar la información y las capacidades disponibles en ODL.

Plataforma De Controladores.- El propósito es aprovechar el modelo de datos SAL(Capa de abstracción de servicios), proporcionar capacidades de SDN fundamentales y funciones de red, tales como la topología, la supervisión del rendimiento, la gestión de conmutador físico o virtual, y la manipulación de tablas ARP. Esta es la capa que actúa como la conexión entre las interfaces hacia el sur y las interfaces hacia el norte.

La capa de servicios de abstracción (SAL).- El objetivo principal de SAL es mapear un conjunto diverso de tecnologías de red en un modelo común de datos abstractos. Todos los servicios de controlador de ODL operan en este modelo abstracto de datos, creando así un controlador SDN neutral. SAL y los diferentes protocolos que se utilizan para comunicarse con los dispositivos de red en conjunto conforman la interfaz de comunicación sur de ODL.

plugins de protocolo.- ODL ha incorporado soporte para diferentes protocolos, mediante el cual se puede comunicar con dispositivos de red. Los plugins más populares son OVSDB, ***OpenFlow y Netconf;*** estos protocolos se utilizan para la gestión de dispositivos de conmutación de capa 2 físicos y virtuales.

Como funciona Opendaylight

El controlador Opendaylight puede implementarse en una variedad de entornos de red, el funcionamiento se da con la interface hacia el norte (nortbound) permite comunicarse con controlador Opendaylight 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 Opendaylight 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 Opendaylight, permite reducir 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 atreves de un API.

Para finalizar el estudio de SDN sintetizamos nuestro análisis afirmando que SDN puede crear una infraestructura que sea mucho más ágil y flexible a través de la automatización de funciones y crear redes programables, y personalizando los requisitos de cada red. Dicha función de programación debe impulsar la automatización y orquestación de red que soporte las demandas de constante cambio de los usuarios, así como los dispositivos y los datos que acceden a la red.

OpenStack

Opensatck es una solución de cloud computing de código abierto y que ha simplificado enormemente el camino de la virtualización. Es una solución flexible para la creación de nubes públicas como privadas, y para ello se consideran dos requerimientos: las nubes deben ser simples de implementar y masivamente escalables La virtualización de las funciones de red (NFV) es ahora sinónimo de OpenStack [19].

OpenStack no solo ha reducido enormemente los costes de despliegue de plataformas gestionadas por la nube (CMP), sino también nos va a simplificar enormemente en la gestión de la infraestructura de red, a través de funciones de red virtuales para la creación de componentes principales de hardware. Estos módulos básicos incluyen: recursos de computación (Nova), redes (Neutrón), almacenamiento (Swift), una interfaz de gestión de usuarios (Horizon), router Neutrón y un repositorio de máquinas virtuales (Glance) [19][20].

De acuerdo a las especificaciones y módulos de OpenStack, la capacidad de lanzar un componente NFV, los usuarios pueden cargar imágenes en Glance, se muestra que Nova está ejecutando la imagen de la máquina virtual (VM), el módulo Neutrón está operativo ya que permite el acceso a la VM instanciada y lanzada; y los routers Neutrón son los enlaces de comunicación entre las instancias mediante enrutamiento.

Arquitectura de OpenStack

OpenStack tiene una arquitectura como se visualiza en la Figura 10, que ofrece un servicio confiable y rentable a la hora de implementar, los principales componentes son [19]:

- **Dashboard:** Nos entrega una interfaz a los clientes y administradores.
- **Compute (Nova):** Genera imágenes y las convierte en instancias o máquinas virtuales.
- **Neutrón:** Provee redes y routers virtuales que son posteriormente asociadas a las instancias con las redes privadas y también acceso desde redes públicas.
- **Block Storage (Cinder):** Permite el almacenamiento de las máquinas virtuales alojadas en la nube.
- **Image (Glance):** Provee una memoria y repositorio para las imágenes.
- **Object Store (Swift):** Permite el almacenamiento de objetos, funciona como un contenedor para el almacenamiento de archivos y recuperarlos posteriormente.
- **Identity (Keystone):** Permite la autenticación y autorización de todos los servicios que tiene la nube.

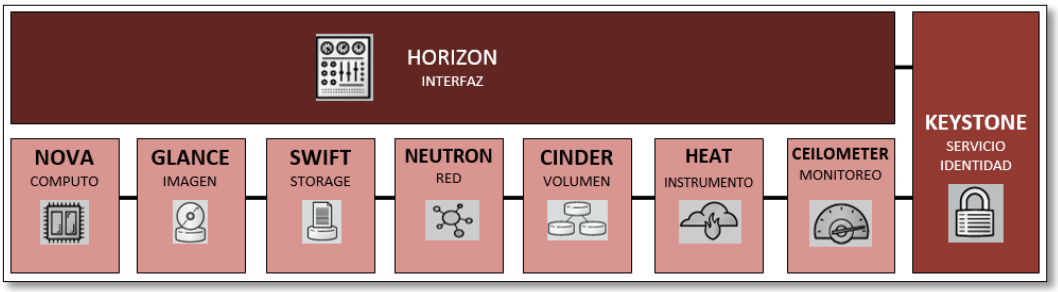

Figura 10 Arquitectura de Openstack

Como funciona Openstack

El funcionamiento de Openstack se basa en tres componentes esenciales, los cuales son los encargados de una comunicación mutua entre sí, estos elementos son las funciones de computación (Compute), redes (Networking) y almacenamiento (Storage).

Los usuarios tienen acceso a una configuración sencilla a través de una interfaz web (Dashboard), en la cual se realiza el inicio de sesión mediante un usuario y contraseña, con el perfil de administrador (admin) podemos crear las redes, subredes, routers, firewalls, máquinas virtuales (instancias) y tener manejo de recursos en la nube.

Openstack ofrece y entrega el control de recursos de la computación, almacenamiento y red a través de un centro de datos, todo es administrado por un API de Openstack. Al trabajar con tecnologías empresariales y código abierto, nos permite la interoperabilidad con infraestructuras heterogéneas en nubes privadas y públicas, lo que origina tener la facilidad de construir y ejecutar aplicaciones que permitan la portabilidad y el aprovechamiento de un gran ecosistema de herramientas y servicios en la nube.

Open Vswitch

Open Vswitch es un conmutador de software cuya función es la gestión de redes virtuales, que se encuentran en ambientes como hipervisores o en la nube, en OpenStack, está integrado a la gestión de redes, por lo que, tanto el nodo Neutrón como el nodo de cálculo (Nova) ejecutan Open vSwitch para proporcionar servicios de red virtualizados. Open Vswitch soporta además la integración con el protocolo Openflow Standard de facto de la redes definidas por Software.

Los plugins de neutrón permiten la generalización y la caracterización de la funcionalidad del neutrón, tales como permitir la interoperabilidad entre OpenStack neutrón y su software y hardware específico. El agente neutrón implementan funcionalidades de red como el agente DHCP y el agente de capa 3. El servidor principal de neutrón y los plugins, se comunican con los agentes de neutrón. La definición de objetos de un API y servicios de red, junto a un plug-in neutrón permite la interoperabilidad entre los routers, switches virtuales y controladores con las redes definidas software (SDN) [21].

Características de Open vswitch

Las características principales a tomar en cuenta que posee OpenvSwitch, se puede citar las siguientes:

- Soporta el protocolo IPv6
- Múltiples protocolos de tunelización, como: GRE, VXLAN, IPsec
- Visibilidad en la Comunicación ente máquinas virtuales mediante Netflow, OpenFlow.
- Configuración de calidad de servicio QoS.
- Reenvío de datos de alto rendimiento mediante el kernel de Linux.

Arquitectura de OpenvSwitch

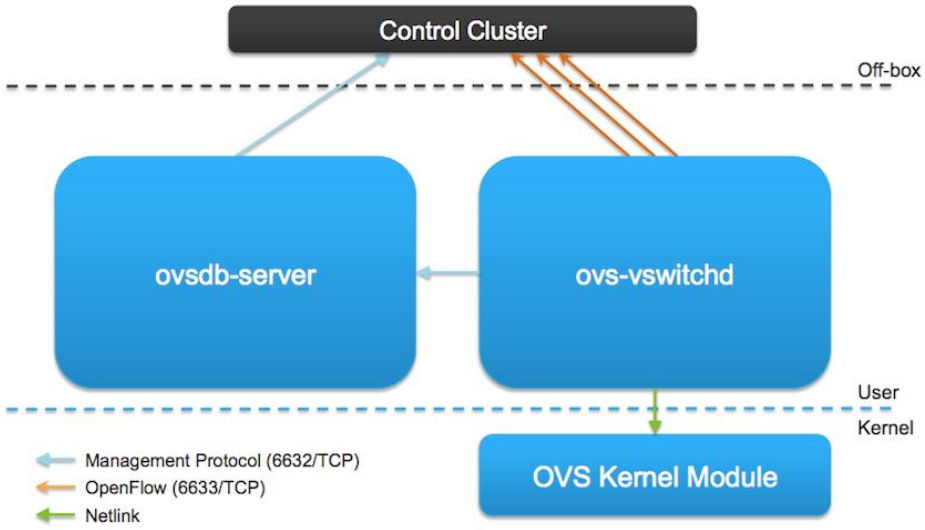

Figura 11 Arquitectura de vSwitch[22]

- ovs-vswitchd.- se encarga de la lógica de reenvio, aprendizaje y vinculación de la configuración remota.
- ovsdb-server.- inicia la conexión del servidor de la base de datos vSwitch
- openvswitch_mod.ko.- módulo kernel (ruta rápida); inicia la búsqueda, modificación y el reenvío de datos mediante el túnel.

Funcionalidades de OpenvSwitch

En un entorno con varios nodos de computación como tenemos en OpenStack, precisamos de algunas funcionalidades como:

- Las máquinas se ejecutan en el mismo nodo de computación, pero que están en diferente red local.
- Las máquinas se ejecutan en diferentes nodos de computación, pero virtualmente están en la misma red local.

- La migración de las máquinas que puedan pasar en vivo “prendida”.
- se tiene servidor DHCP por cada subred que no interfiera con los de otras subredes.
- Reglas de firewalls están definidas por máquina.
- El tráfico de red es seguro entre nodos de computación.
- Realiza la conexión de las máquinas virtuales con el mundo exterior.

Virtualización de las funciones de la red NFV

La virtualización de las funciones de red (NFV) ofrece a los proveedores de servicios de red la libertad de elección y permite separar el software de red del hardware. Este desacoplamiento brinda ventajas tales como el ahorro de costos en la implementación y el funcionamiento de la red, el aprovisionamiento rápido a pedido de nuevas funciones de red, una mayor eficiencia y una escalabilidad de red ágil. Estas ventajas abren la puerta a nuevas oportunidades comerciales, brindan nuevos servicios al mercado con mayor rapidez y han despertado un enorme interés entre los proveedores de servicios de Internet y en la nube, los operadores de telefonía móvil y los segmentos del mercado empresarial [10].

NFV se basa en el concepto de virtualización de servidores. Se expande el concepto más allá de los servidores, ampliando el alcance para incluir dispositivos de red. También permite que el ecosistema administre, aprovisione, supervise e implemente estas entidades de red virtualizadas. Ofreciendo soluciones prometedoras para simplificar, optimizar y transformar la red en una red rentable y flexible. Esta transformación de una red a NFV es más que solo una transición de física a virtual. Necesita un cambio de paradigma en la forma en que se construyen las redes. Los principios del diseño de red tradicional deben ser aumentados con los principios de diseño para NFV, como la ubicación de la función de red virtualizada (VNF) y el diseño de la infraestructura. [10]

Consideraciones de Diseño para NFV

En la figura 12 podemos ver las consideraciones de diseño para NFV

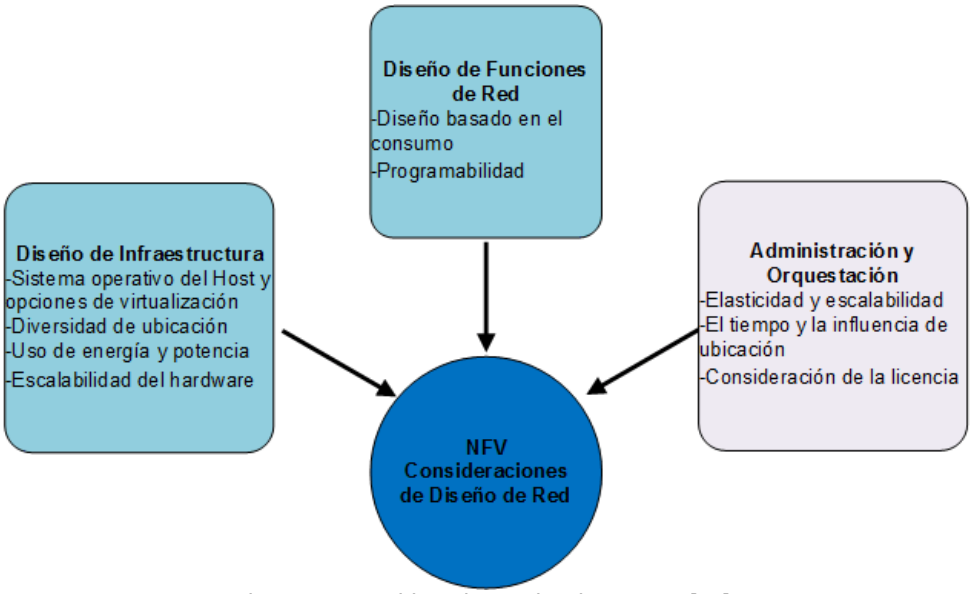

Figura 12 Consideraciones de Diseno NFV[10]

Diseño de infraestructura de NFV

La infraestructura para el NFV no está diseñada para satisfacer las necesidades de una red y servicios específicos. La infraestructura debe ser genérica y debe garantizar que pueda permitir la escalabilidad y elasticidad de las NFv. Las siguientes características deben ser analizadas para el diseño de una solución NVF

Recursos de hardware escalables

El hardware de la infraestructura debe tener flexibilidad para escalar hacia arriba y hacia abajo si es necesario. Al no depender únicamente de la red subyacente es necesario establecer un crecimiento planificado.

Costos de hardware y gastos de capital

El costo del hardware siempre es un criterio de selección importante. Se percibe que el hardware personalizado es la mejor manera de alcanzar el punto de precio óptimo para el hardware. Por lo que es necesario que los dispositivos sean compatibles y que el retorno de la inversión se vea sostenible y sustentable en el ciclo de vida útil de una infraestructura NFV.

Elección del sistema operativo host y la capa de virtualización. La selección de del hypervisor debe estar alineado con el sistema operativo, los dos deben ser compatibles, sin embargo, pueden eliminar los costos de licenciamiento y confiar en estructuras de soporte internas, de terceros o basadas en la comunidad para el crecimiento futuro y la resolución de problemas.

Diseño de las funciones de red.

Un diferenciador clave con NFV es que el diseño y la implementación adoptan un enfoque centrado en el software para que la funcionalidad principal de la red se implemente en el software. Por lo tanto, las funciones de red (VNF) pueden agregarse, ampliarse y reducirse, eliminarse y reubicarse puramente en software.

Administración y Orquestación.

También se espera que los NFV admitan API abiertas, lo que permite que cualquier herramienta de gestión y orquestación de terceros controle la ubicación y los ciclos de vida de los VNF. La herramienta de orquestación instruye a la capa de virtualización a interconectar el VNF en el orden deseado, y los nuevos VNF se pueden crear instancias y agregar a los datos o controlar la ruta del tráfico sobre la marcha. [10].

Las características expuestas de SDN y NVF nos permite entender que son dos funcionalidades complementarias y no dependientes, ya que el despliegue de la primera no obliga a la implementación de la segunda.

Además, que cuentan con aspectos comunes como son:

Mover la funcionalidad al software

Aproveche las interfaces del programa de aplicación (API)

Admite orquestación, virtualización y automatización de servicios de red más eficientes.

Capítulo 4

Marco Metodológico

Metodología

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 de Red

La topología que se va a implementar para las pruebas es una red estrella, sus principales ejes en el desarrollo son las Redes Definidas por Software y Funciones de Red Virtualizadas en un ambiente virtual. Previamente se debe seguir los pasos del Anexo 1.

Topología de la conexión entre el controlador SDN y Mininet

Se procede a utilizar un controlador OpenDayLight, es implementado como se detalla en el Anexo 2. Este controlador se interconecta mediante Openflow el cual es un protocolo que permite el intercambio de datos entre el controlador y los swicth, que son emulados mediante mininet que es un software descargado como se muestra en el Anexo 3, en la cual simula una red que se crea mediante un script realizado en python.

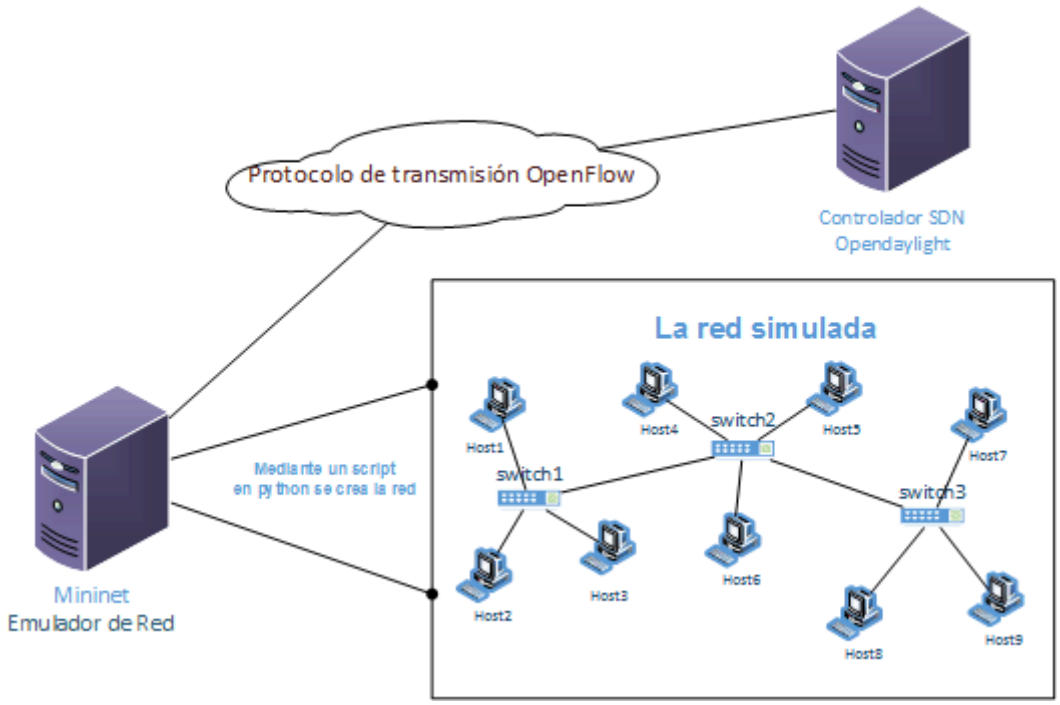

Figura 13 Conexión entre el controlador SDN y Mininet

La red simulada funciona en el protocolo IPv4 e IPv6, para la comunicación se realiza mediante el protocolo de transición DualStack, en donde el cliente (Host) está configurado con las dos direcciones IP. Como se observa en la Figura 14.

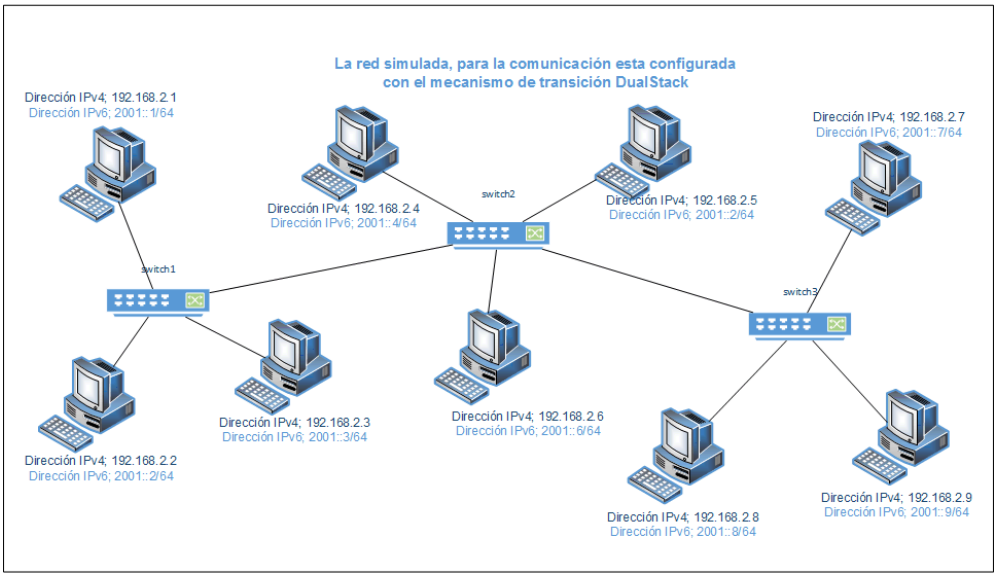

Figura 14 Topología de red emulada en MIninet

Topología de red en la nube (Openstack)

Openstack es un proyecto de cloud computing, su instalación y configuración es realizada en el Anexo 4. Esta topología consta de la creación de un neutrón, redes que serán asignadas a las instancias que se crearon, estas instancias fueron pre-configuradas con servicios de red. Como se visualiza en la Figura 15.

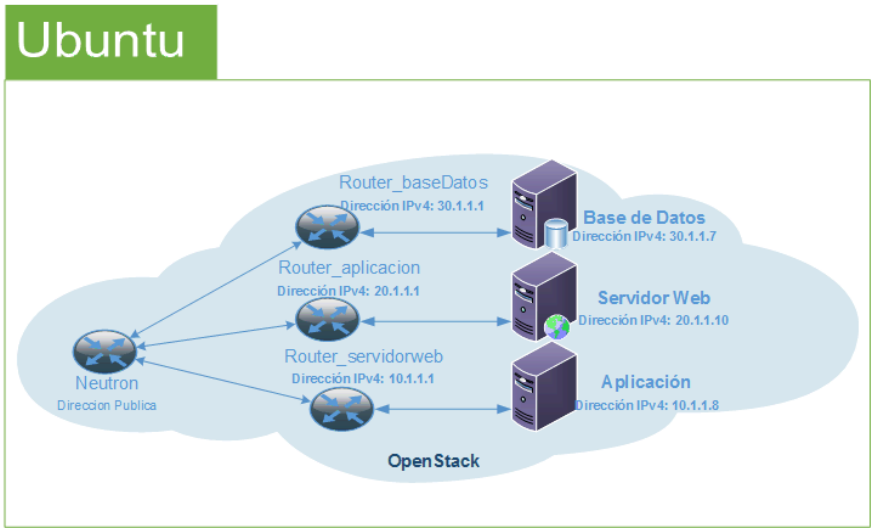

Figura 15 Topología de red en la nube (Openstack)

La topología de red entre Openstack y el controlador Opendaylight, se la realiza mediante el Openflow que permite el intercambio de datos entre el controlador y el router neutrón. En la cual mediante el enrutamiento estático y mediante flujos de datos, se obtiene una conexión entre el neutrón y los servidores de base de datos, web y la aplicación.

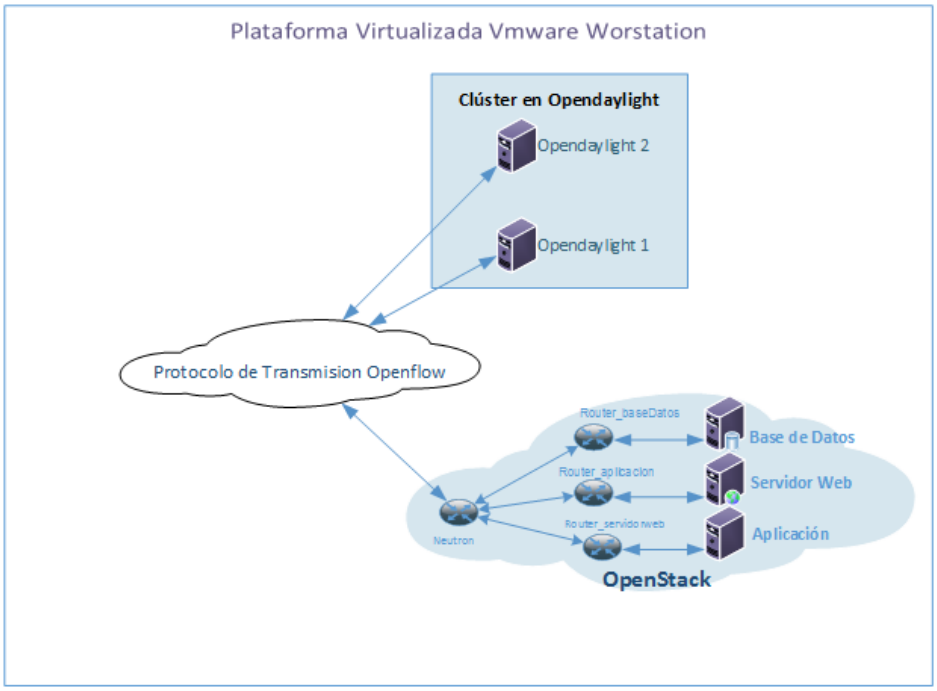

Figura 16 Topología de Openstack y SDN

La topología implementada y desarrollada está definida, como se observa en la Figura 17.

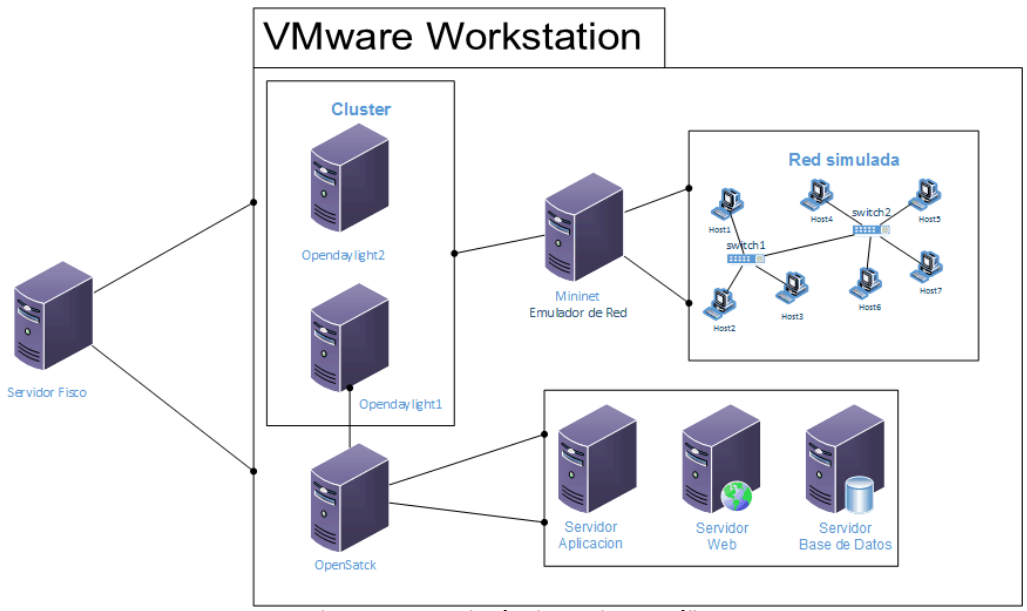

Figura 17 Topología de Red Específica

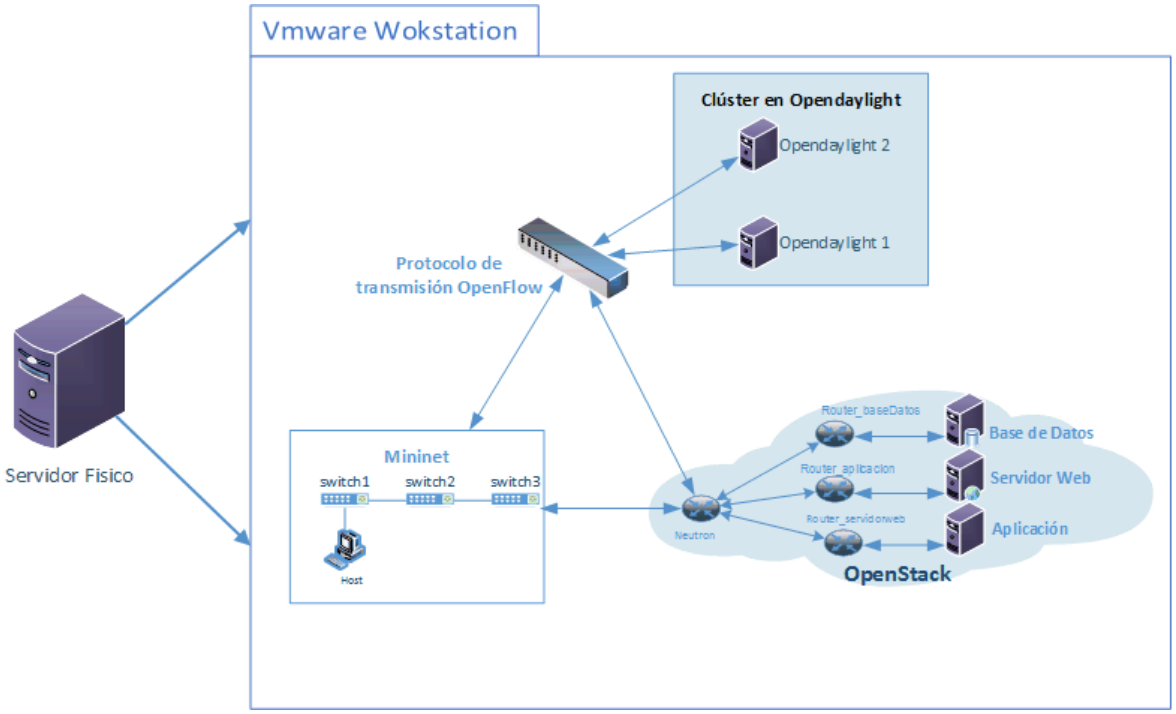

Figura 18 Topología de Red Detallada

Implementación

Teniendo definida la topología y las herramientas que se van a utilizar, se procede a la implementación del proyecto con diversos escenarios que permitirán entender el funcionamiento de cada una.

Para comenzar se realizara la creación de las redes definidas por software (SDN), se utilizara el emulador Mininet, el cual mediante el lenguaje de programación python permite el desarrollo de scripts para crear topologías de red.

Además se utilizara Openstack, que permitirá realizar un escenario real en la nube que está conformado por redes, subredes, routers, reglas de firewall y máquinas virtuales (instancias).

Finalmente para realizar las pruebas se propone utilizar la herramienta Wireshark, que es un analizador de tráfico en el cual permitirá visualizar los flujos de datos que se están transmitiendo en la red SDN.

Pruebas

Primer escenario

Se realizó con el controlador Opendaylight (ODL), conectado dos switches y cada nodo contiene cinco hosts. Tal como se muestra en la Figura 19.

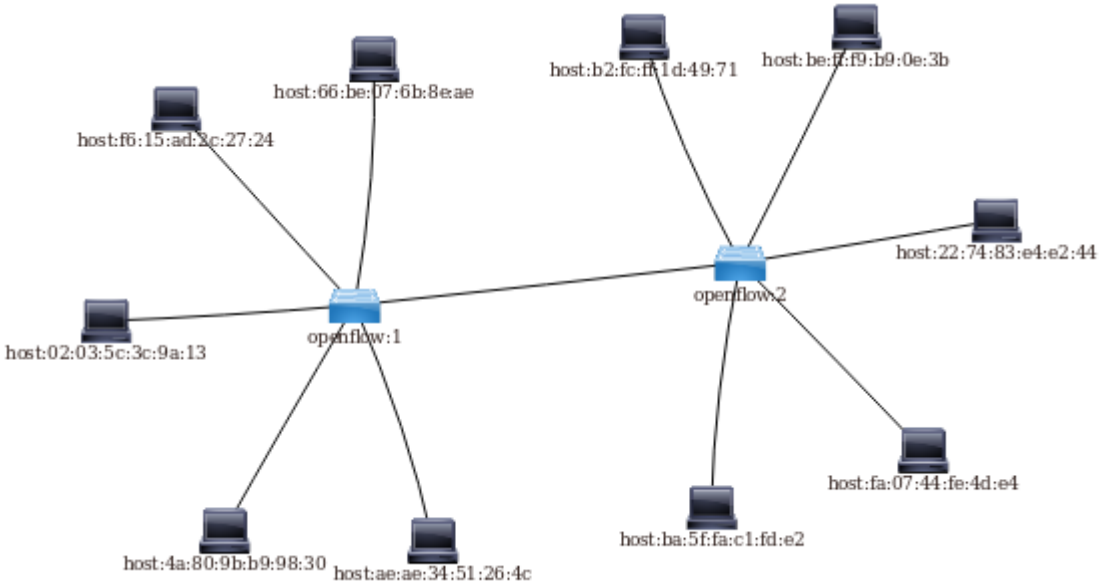

Figura 19 Topología dos switches con cinco hosts cada uno

Para la generación de la topología de prueba inicialmente se despliega el script de python en el emulador Mininet, éste script contiene toda la configuración y creación de cada componente de red.

```
mininet> dump
<Host h1: h1-eth0:10.1.1.1 pid=99531>
<Host h2: h2-eth0:10.1.1.2 pid=99535>
<Host h3: h3-eth0:10.1.1.3 pid=99537>
<Host h4: h4-eth0:10.1.1.4 pid=99539>
<Host h5: h5-eth0:10.1.1.5 pid=99541>
<Host h6: h6-eth0:10.1.1.6 pid=99543>
<Host h7: h7-eth0:10.1.1.7 pid=99545>
<Host h8: h8-eth0:10.1.1.8 pid=99547>
<Host h9: h9-eth0:10.1.1.9 pid=99549>
<Host h10: h10-eth0:10.1.1.10 pid=99551>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None,s1-eth4:None,s1-eth5:None,s1-eth6:None pid=99556>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None,s2-eth4:None,s2-eth5:None,s2-eth6:None pid=99559>
<RemoteController c0: 127.0.0.1:6633 pid=99525>
<RemoteController c1: 192.168.1.111:6633 pid=99725>
```

Figura 20 Detalle de la topología creada con 2 switches

En la Figura 20, se observa los hosts con su interfaz de red y su dirección IP; la conectividad con el controlador se la realiza mediante los flujos de datos agregados en cada switch.

Pruebas de Conectividad

Se puede observar en las Figuras 21 y 22, que se tiene los hosts configurado la dirección IP. En este caso se tiene host1 y al host8, se visualiza que se tiene conectividad entre los hosts especificado.

Figura 21 Host1 con la dirección IPv4

Figura 22 Host8 con la dirección IPv4

Segundo Escenario

Está conformado por los siguientes componentes: un controlador Opendaylight (ODL), cinco switches y cada nodo tiene cuatro hosts.

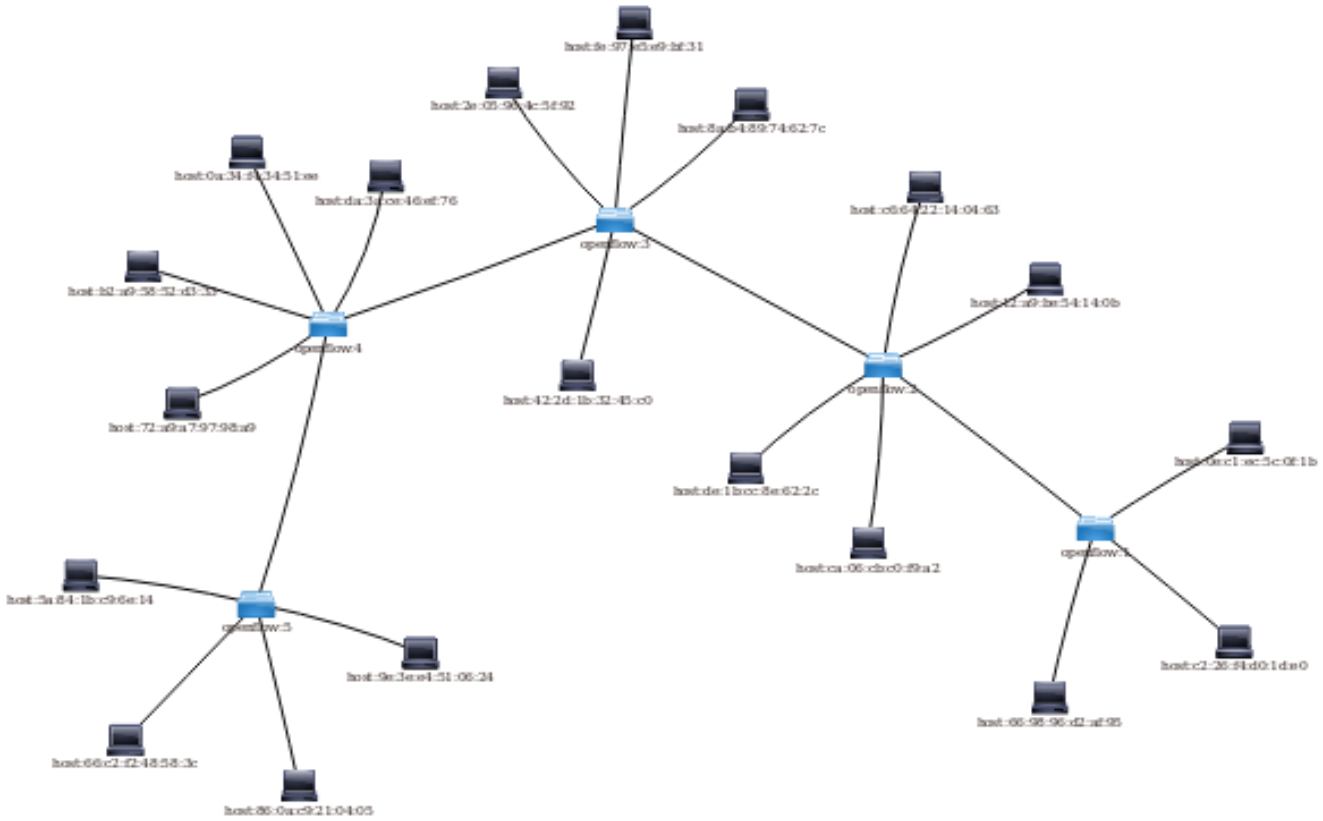

Figura 23 Topología cinco switches con cuatro hosts cada uno

Para la generación del segundo escenario esta topología de prueba se despliega igualmente con un script de python en Mininet, éste script se realizó la modificación para la colocación de cinco switches y en cada nodo está conectado cuatro máquinas (hosts) con direcciones IPv6.

```
mininet> dump
<Host h1: h1-eth0:0.0.0.0 pid=3346>
<Host h2: h2-eth0:0.0.0.0 pid=3350>
<Host h3: h3-eth0:0.0.0.0 pid=3352>
<Host h4: h4-eth0:0.0.0.0 pid=3354>
<Host h5: h5-eth0:0.0.0.0 pid=3356>
<Host h6: h6-eth0:0.0.0.0 pid=3358>
<Host h7: h7-eth0:0.0.0.0 pid=3360>
<Host h8: h8-eth0:0.0.0.0 pid=3362>
<Host h9: h9-eth0:0.0.0.0 pid=3364>
<Host h10: h10-eth0:0.0.0.0 pid=3366>
<Host h11: h11-eth0:0.0.0.0 pid=3368>
<Host h12: h12-eth0:0.0.0.0 pid=3370>
<Host h13: h13-eth0:0.0.0.0 pid=3372>
<Host h14: h14-eth0:0.0.0.0 pid=3374>
<Host h15: h15-eth0:0.0.0.0 pid=3376>
<Host h16: h16-eth0:0.0.0.0 pid=3378>
<Host h17: h17-eth0:0.0.0.0 pid=3380>
<Host h18: h18-eth0:0.0.0.0 pid=3382>
<Host h19: h19-eth0:0.0.0.0 pid=3384>
<Host h20: h20-eth0:0.0.0.0 pid=3386>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None,s1-eth4:None,
s1-eth5:None pid=3391>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None,s2-eth4:None,
s2-eth5:None,s2-eth6:None pid=3394>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None,s3-eth3:None,s3-eth4:None,
s3-eth5:None,s3-eth6:None pid=3397>
<OVSSwitch s4: lo:127.0.0.1,s4-eth1:None,s4-eth2:None,s4-eth3:None,s4-eth4:None,
s4-eth5:None,s4-eth6:None pid=3400>
<OVSSwitch s5: lo:127.0.0.1,s5-eth1:None,s5-eth2:None,s5-eth3:None,s5-eth4:None,
s5-eth5:None pid=3403>
<RemoteController c0: 127.0.0.1:6633 pid=3340>
<RemoteController c1: 192.168.1.111:6633 pid=3727>
mininet>
```

Figura 24 Detalle de la topología creada en IPv6

En la Figura 24, se observa la información detallada de la estructura de las máquina (hosts) en este caso por tener direcciones IPv6, no nos vizualia las direcciones IPv4; los flujos de datos creados se encargan de realizar la conexión con cada uno de los switches y con el controlador Opendaylight (ODL).

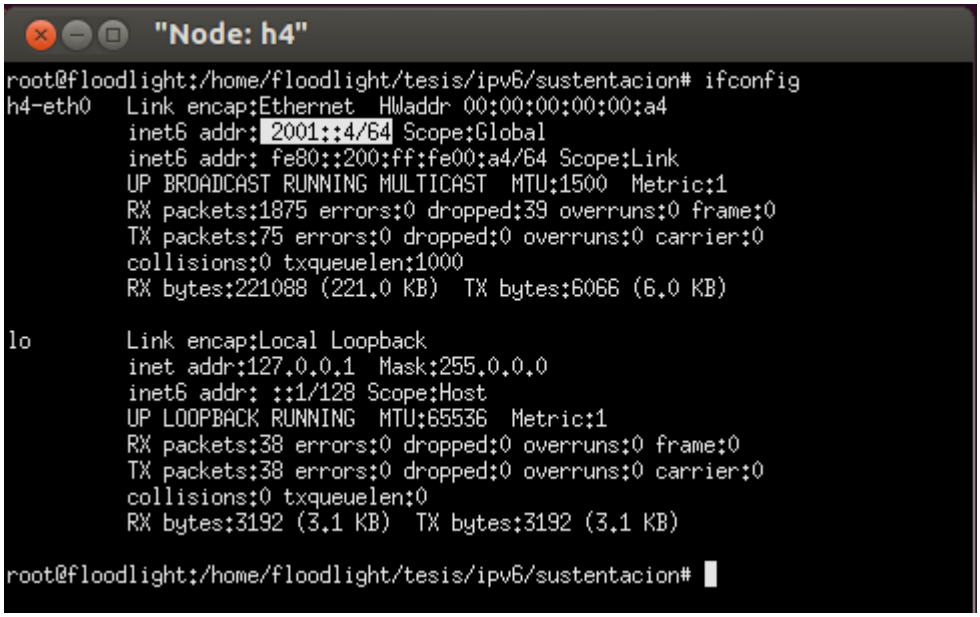

Figura 25 Direccion IPv6 del Host

En la Figura 25, mediante el comando ifconfig se observa en el host4 que esta configurado con la dirección IPv6.

Conectividad ICMPv6 (ping6)

Posteriormente mediante la herramienta de wireshark en el cual mediante el protocolo ICMPv6 se podrá observar el tráfico en los nodos especificados:

La prueba se realizó con los nodos que tienen las siguientes direcciones IP:

- Host11: 2001::11 conectividad con la dirección host7: 2001::7
- Host2: 2001::2 conectividad con la dirección host17: 2001::2

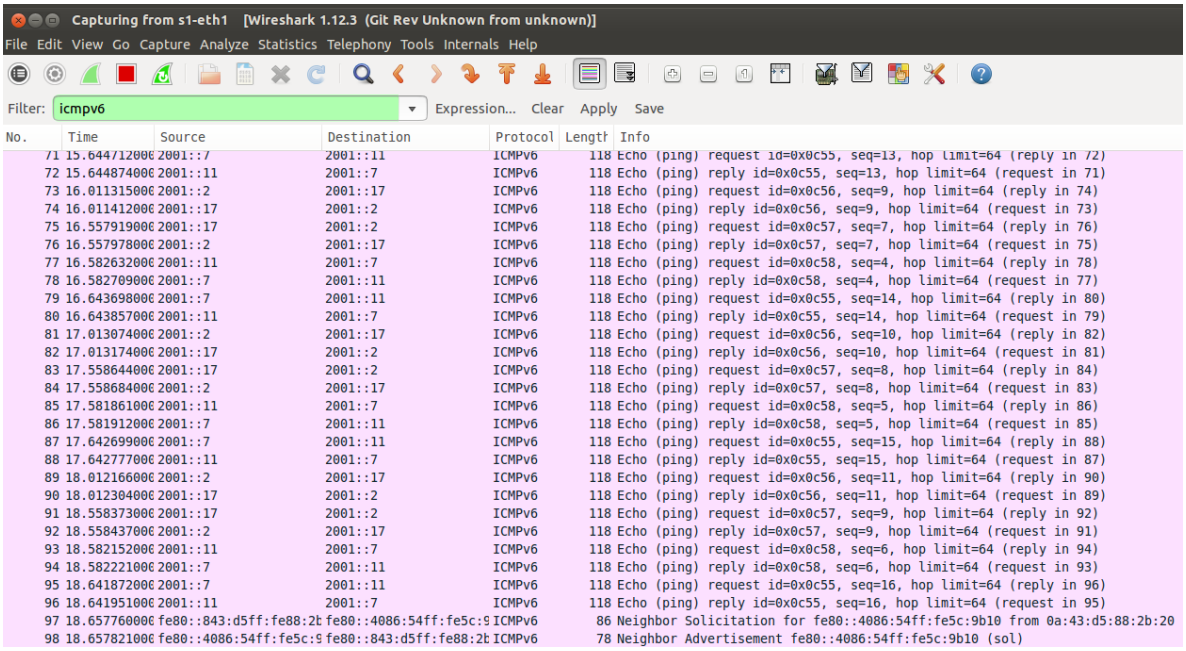

Figura 26 Conectividad en los hosts IPv6

La conexión configurada con el protocolo IPv6 en las máquinas (hosts) con sus direcciones IPv6, se realizó mediante el comando “ping6”, entre las maquinas especificadas da como resultado la conectividad correctamente en la red SDN configurada y creada. Como se observa en la Figura 26.

Tercer escenario

Está con tres switches y con dos hosts, permite la interconexión entre dos hosts de diferentes redes. La primera 192.168.10.0/24 y la segunda es 192.168.50.0/24.

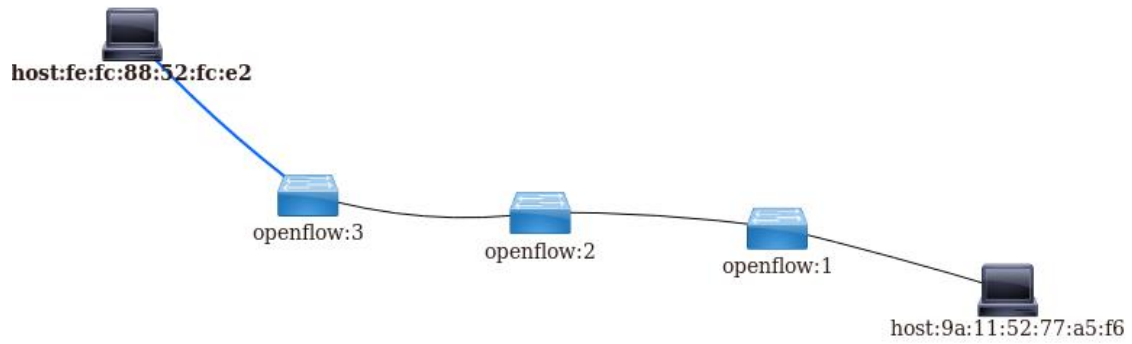

Figura 27 Topología de red escenario3

Se utilizó un segundo controlador llamado pox, el cual permite crear gateways virtuales para la conexión de diferentes redes y a su vez permite la configuración automática de los flujos. Tal como se muestra en la Figura 28.

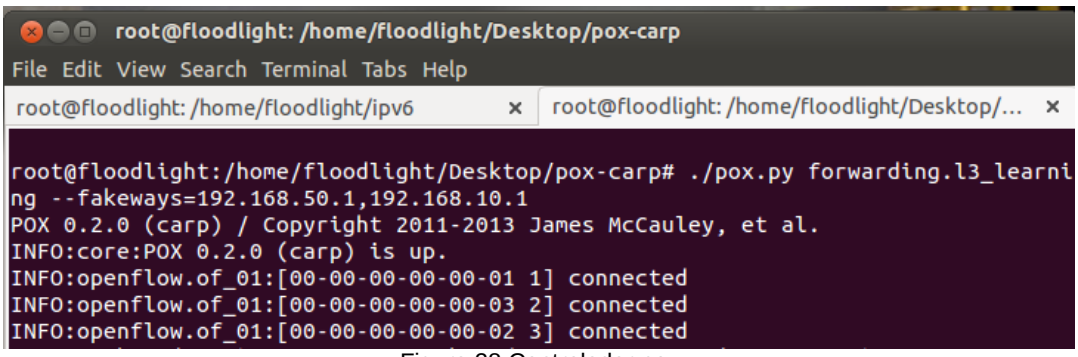

Figura 28 Controlador pox

Mediante el comando dump, se visualiza detalladamente como están los hosts con qué dirección está configurada, los enlaces y la conexión con el controlador. Como se muestra en la Figura 29.

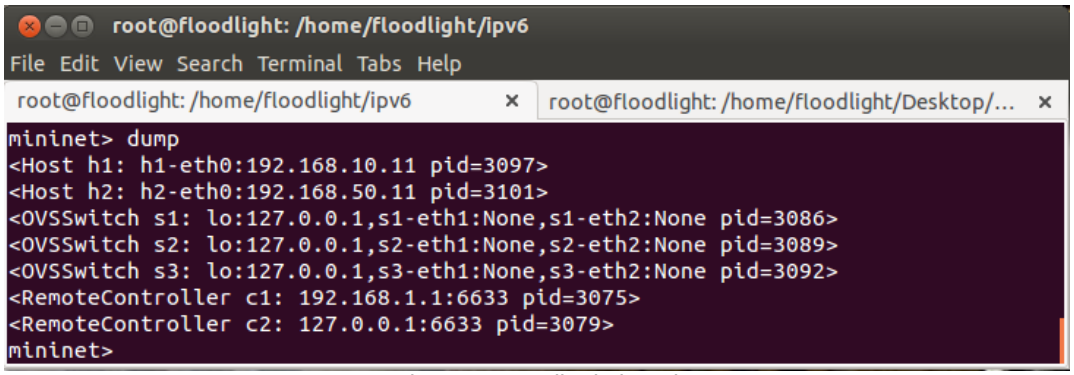

Figura 29 Detalle de la red

Prueba de conectividad

Se puede comprobar que existe conectividad entre diferentes segmentos, teniendo una tabla de enrutamiento entre el host 1 y el host2. Tal como se observa en las Figuras 30 y 31.

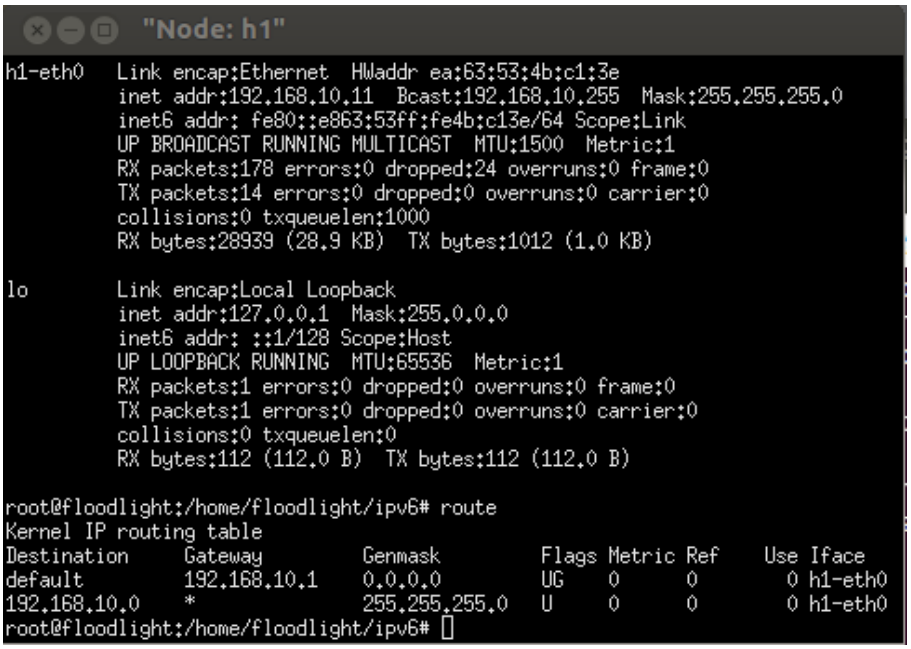

Figura 30 Enrutamiento en el Host1

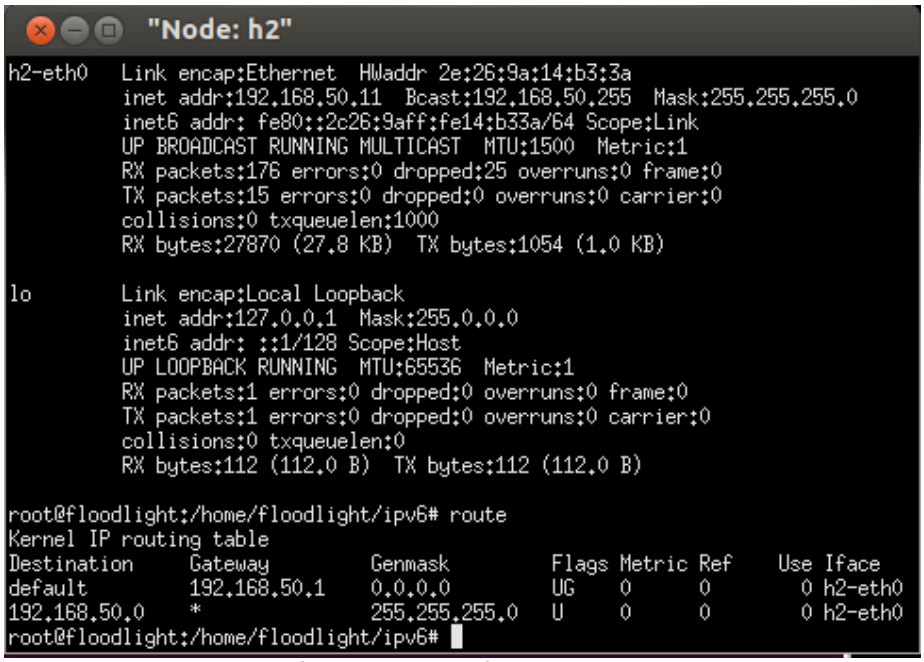

Figura 31 Enrutamiento en Host2

Mediante el comando ping se pudo comprobar la conectividad entre los dos hosts. Con se observa en las Figuras 32 y 33.

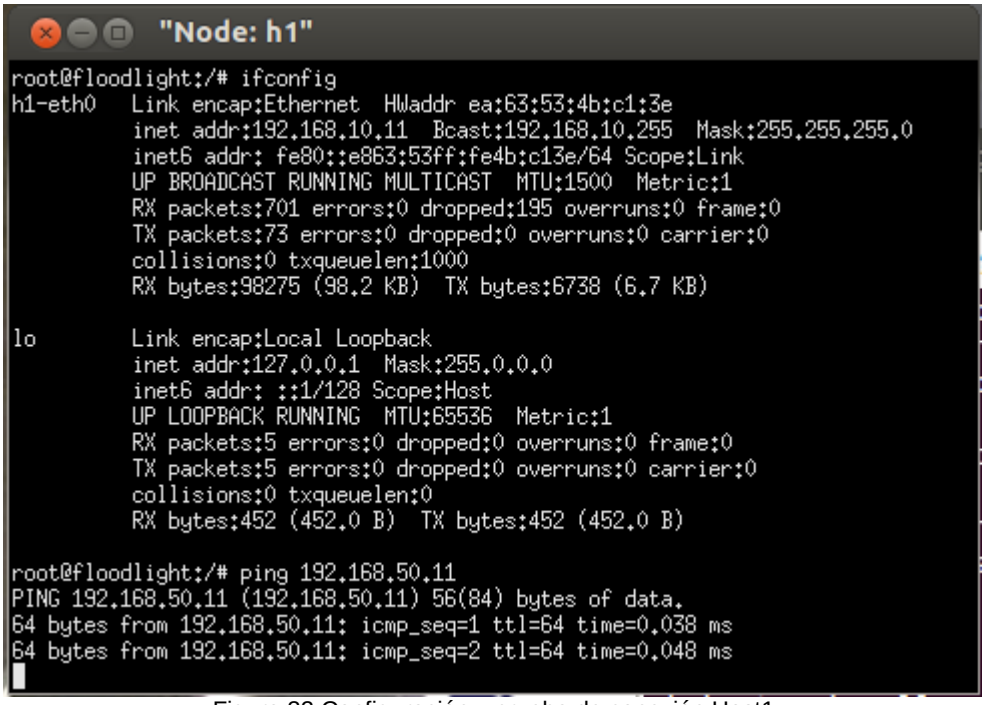

Figura 32 Configuración y prueba de conexión Host1

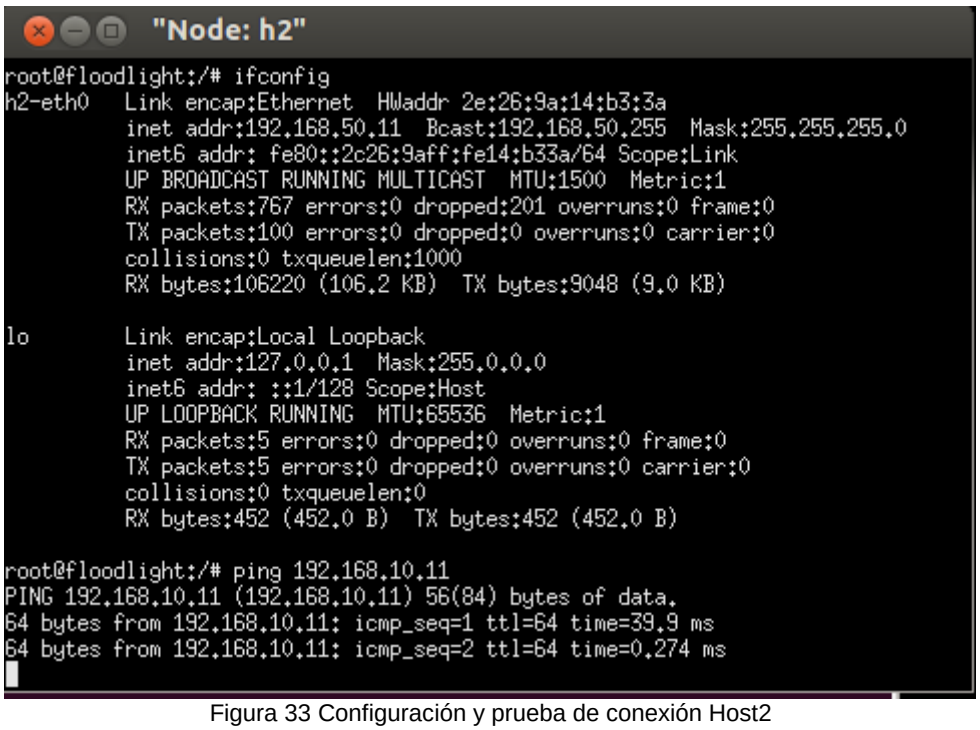

Figura 33 Configuración y prueba de conexión Host2

Cuarto escenario

Está configurado con el mecanismo dual stack, este consta de veinte hosts y cinco switches.

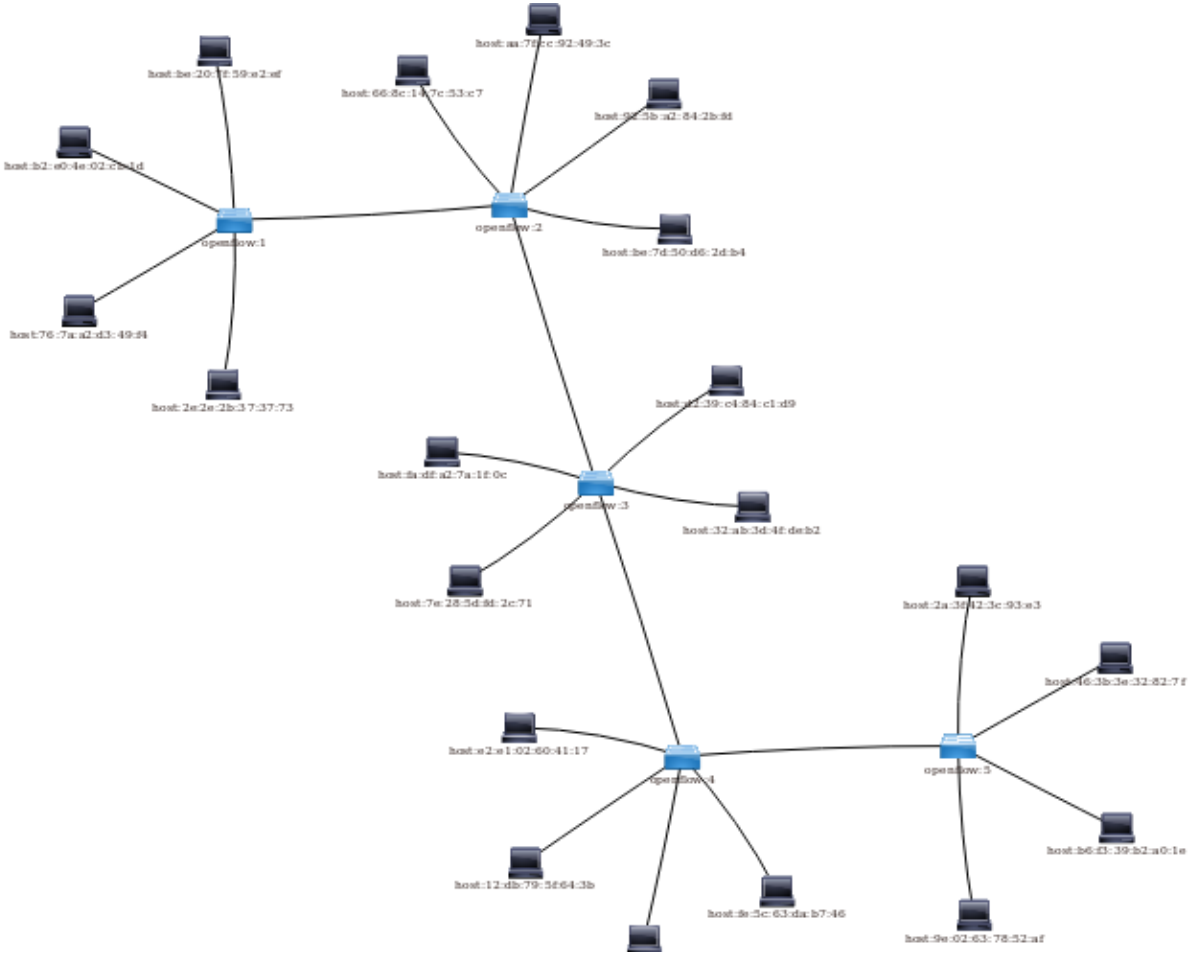

Figura 34 Topología de red escenario 4

Mediante el comando dump, se puede visualizar el detalle de la red. Se observa los veinte hosts, los enlaces de cada switch implementados y la dirección IP del controlador ODL. Como se muestra en la Figura 35.

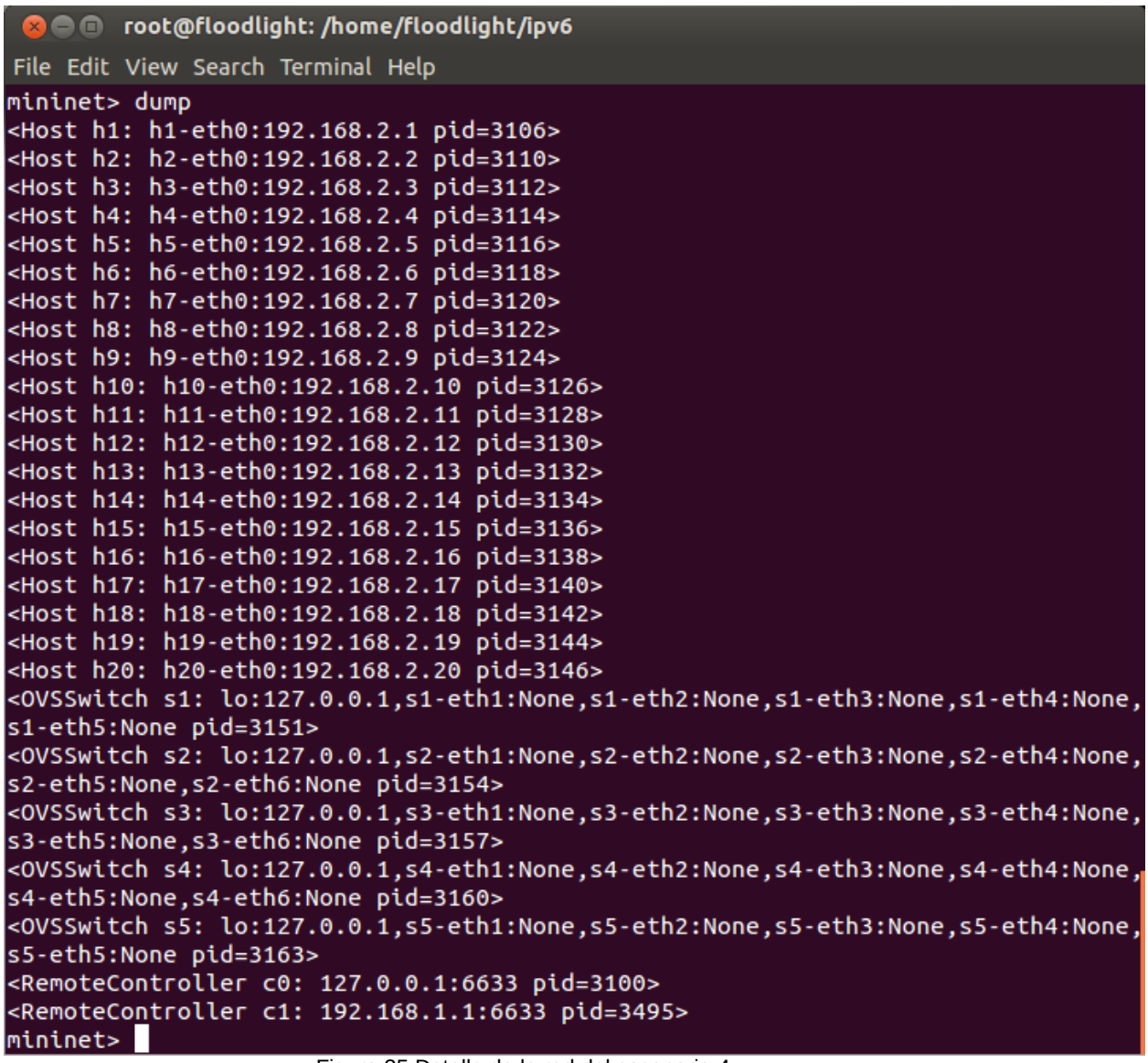

Figura 35 Detalle de la red del escenario 4

En la terminal de mininet, se procedió a comprobar la conexión de la topología de red mediante el comando “pingall”. Como se muestra en la Figura 36, la conexión con cada host fue satisfactoria.

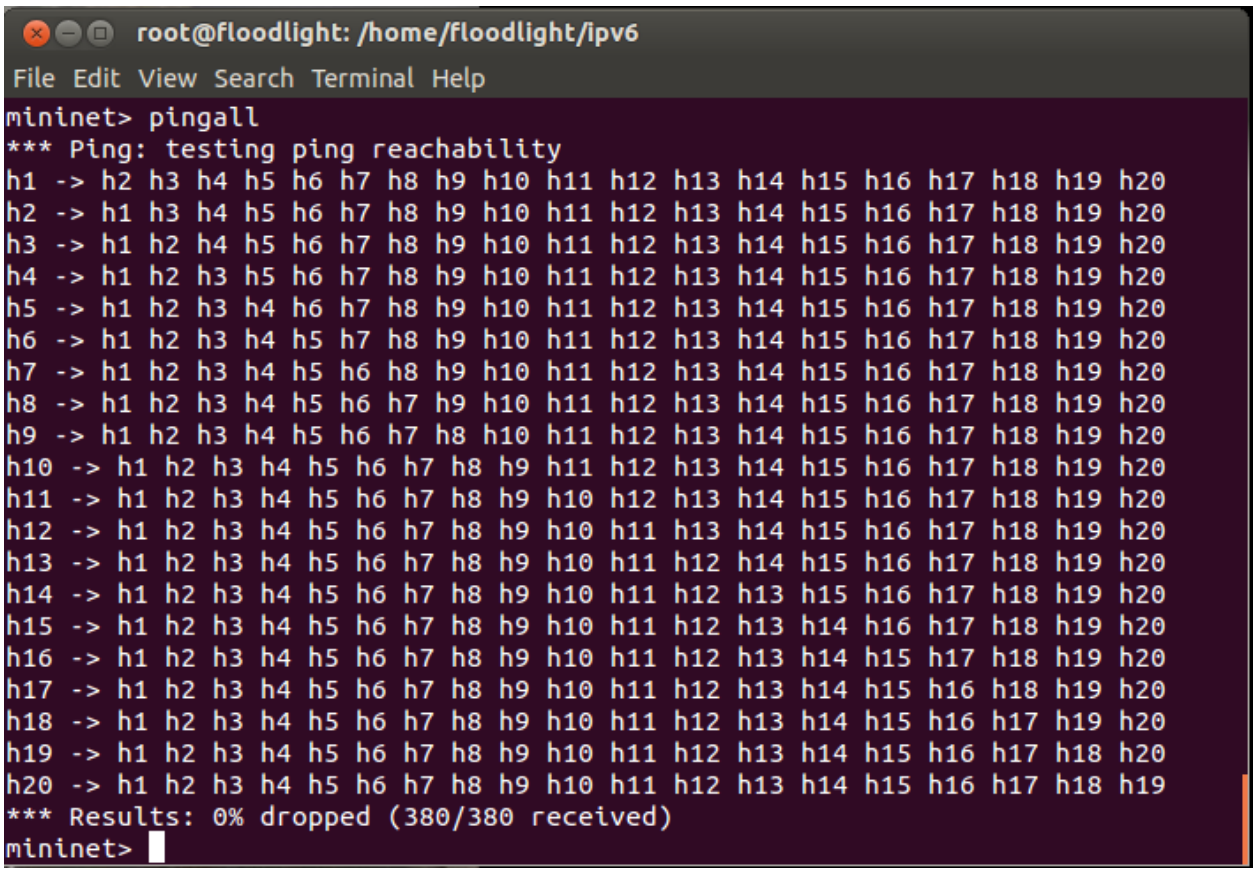

Figura 36 Conectividad de la red en escenario 4

Configuración de los hosts

En las figuras 37 y 38, se puede observar la configuración de la dirección IPv4 e IPv6 en los hosts 1 y 20.

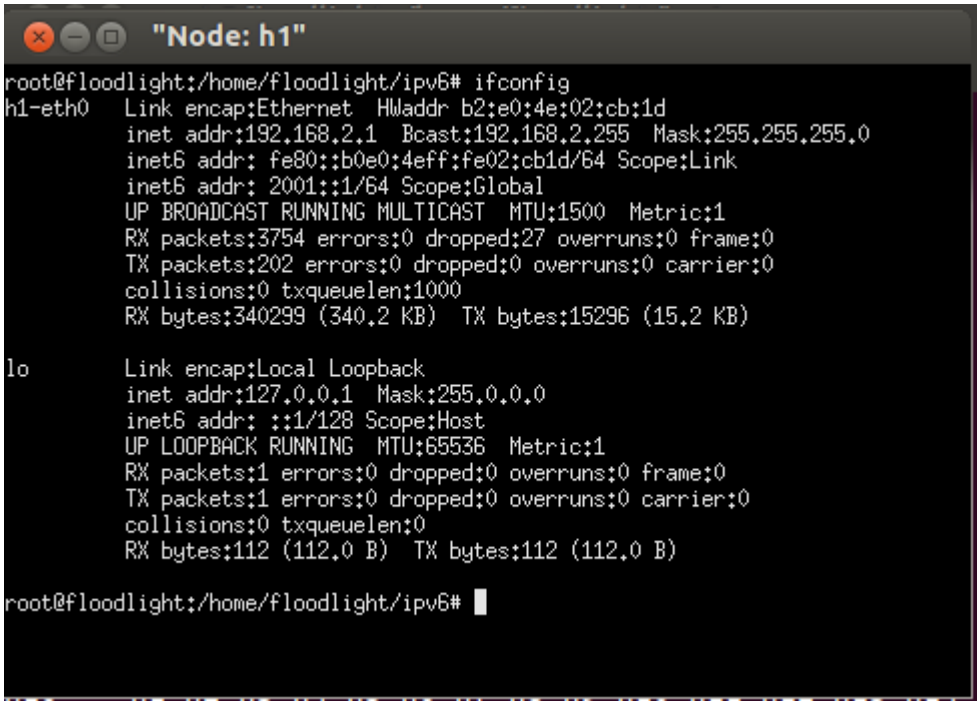

Figura 37 Configuración del Host1

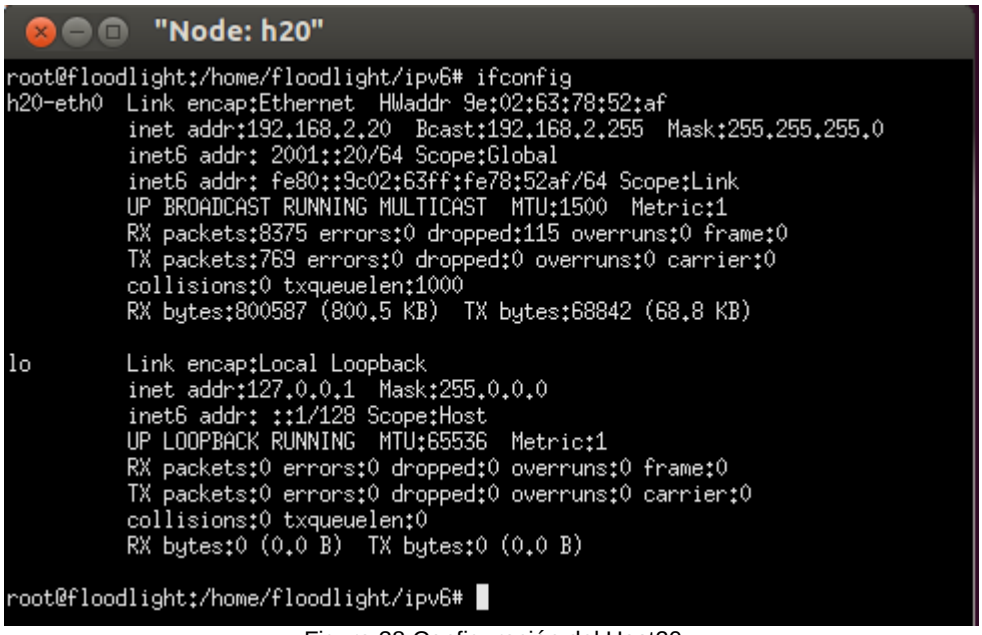

Figura 38 Configuración del Host20

Pruebas de conectividad

Mediante el comando ping se pudo verificar la interoperabilidad entre los dos protocolos de IPv4 e IPv6.

En las Figuras 39 y 40 se visualiza la comprobación entre los dos hosts en el protocolo Ipv4.

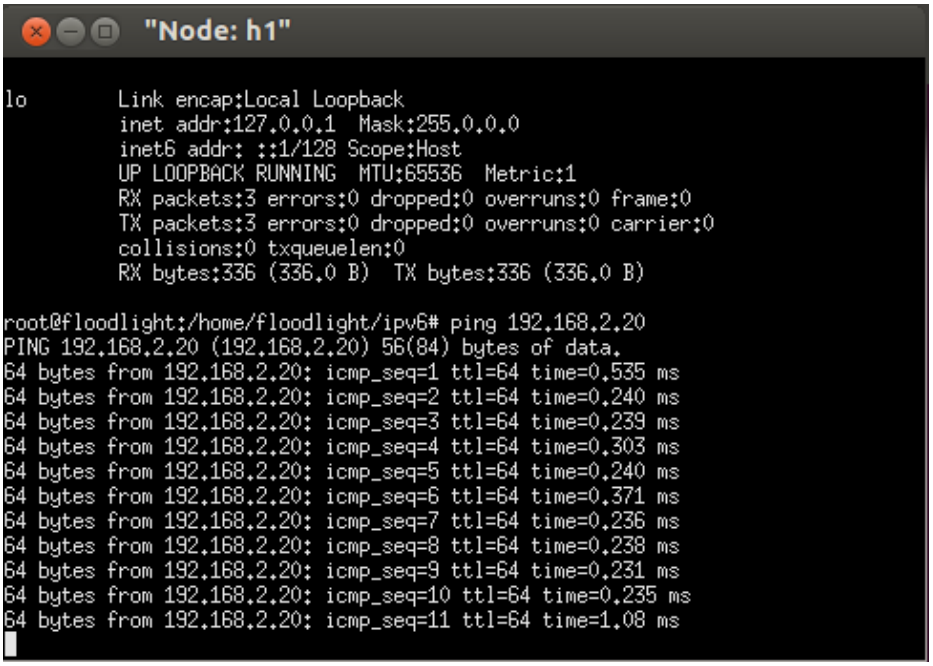

Figura 39 Verificación del Host1 con el protocolo IPv4

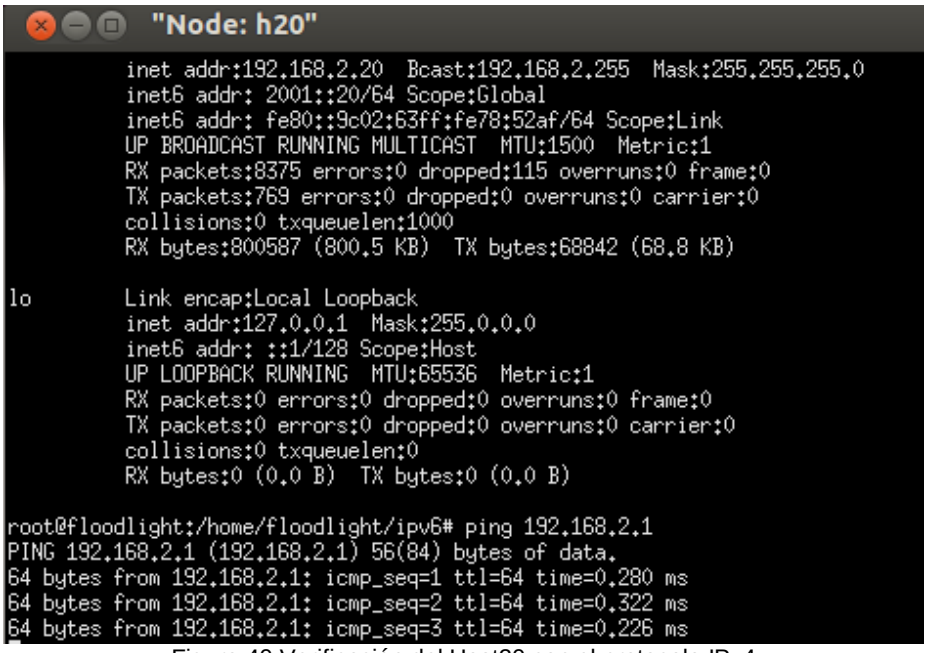

Figura 40 Verificación del Host20 con el protocolo IPv4

En las Figuras 41 y 42 se visualiza la comprobación entre los dos hosts en el protocolo Ipv6.

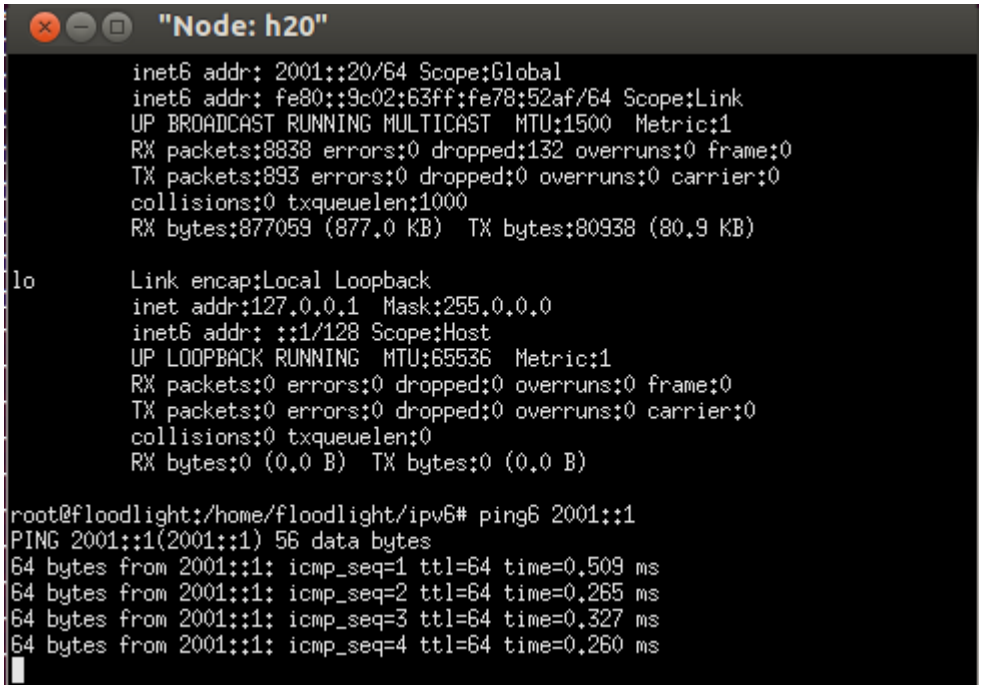

Figura 41 Verificación del Host1 con el protocolo IPv6

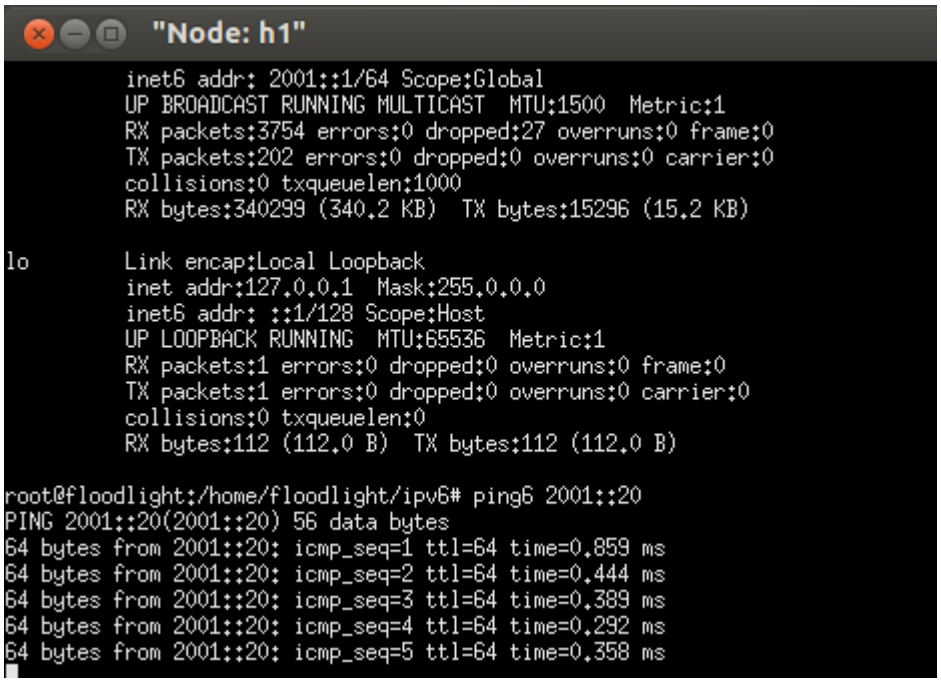

Figura 42 Verificación del Host20 con el protocolo IPv6

Integrar OpenStack-Neutrón con el controlador OpenDayLight

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

1.- Reiniciar el servicio openvswitch

Lo primero que se debe hacer es detener y subir el servicio de openvswitch, para ello son los siguientes comandos:

- *sudo service openvswitch-switch stop*
- *sudo service openvswitch-switch start*

2.- Configuración del puente

Posteriormente se procede a realizar un puente hacia la red interna de OpenStack, que se enlace con la segunda interfaz de Ubuntu con el siguiente comando:

- *sudo ovs-vsctl add-port br-int ens38*

En el cual “br-int” es el nombre de la interfaz interna del router neutrón, “ens38” es el nombre de la segunda interfaz de la máquina virtual de Ubuntu.

3.- Crear conexión para el controlador ODL

Para finalizar se debe crear una conexión con el controlador OpenDayLight de la siguiente manera:

- *sudo ovs-vsctl set-manager tcp:192.168.1.111:6640*

Donde “192.168.1.111” es la dirección IP del controlador Opendaylight y el puerto “6640” es por el cual se comunicará el Neutrón con el controlador.

Luego de haber realizado el paso anterior, se podrá verificar su correcta configuración con el siguiente comando:

- *sudo ovs-vsctl show*

Se puede observar en la Figura 43, la configuración con el router neutrón:

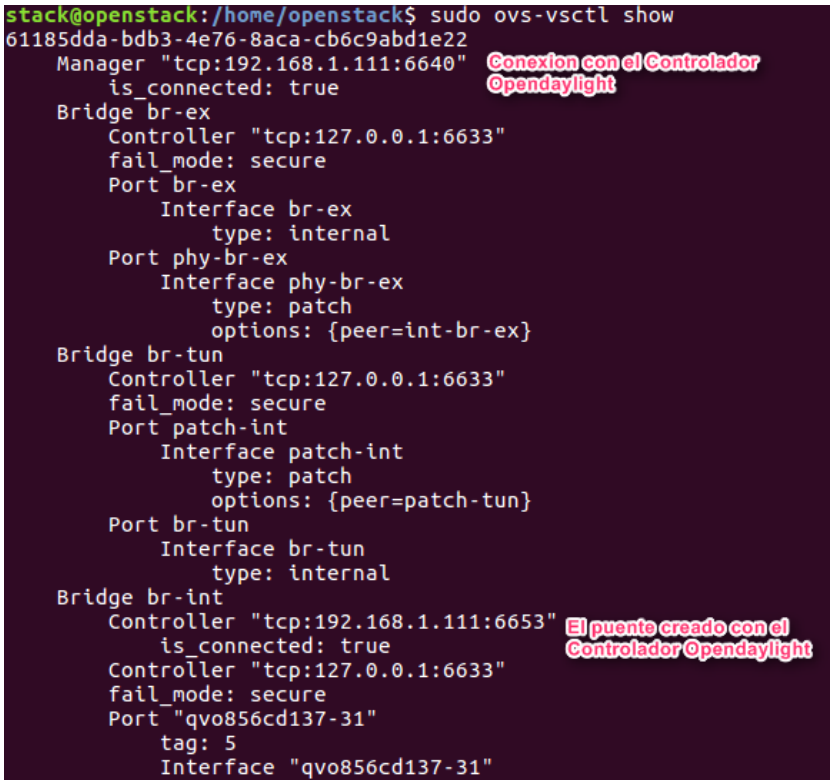

Figura 43 Configuración con el router Neutron

En el controlador Opendaylight, luego de haber agregado el módulo de openvswitch la topología creada con los dispositivos virtuales, en la interfaz de ODL se observara la conexión como se muestra en la figura 44.

En el diagrama de red se puede visualizar los cinco switches, aquellos representan los dispositivos openflow:1, openflow:2, openflow:3 y openflow:4 que son los virtualizados por el script ejecutado en Mininet, mientras que el dispositivo openflow:52226282686 pertenece al router neutrón de Openstack.

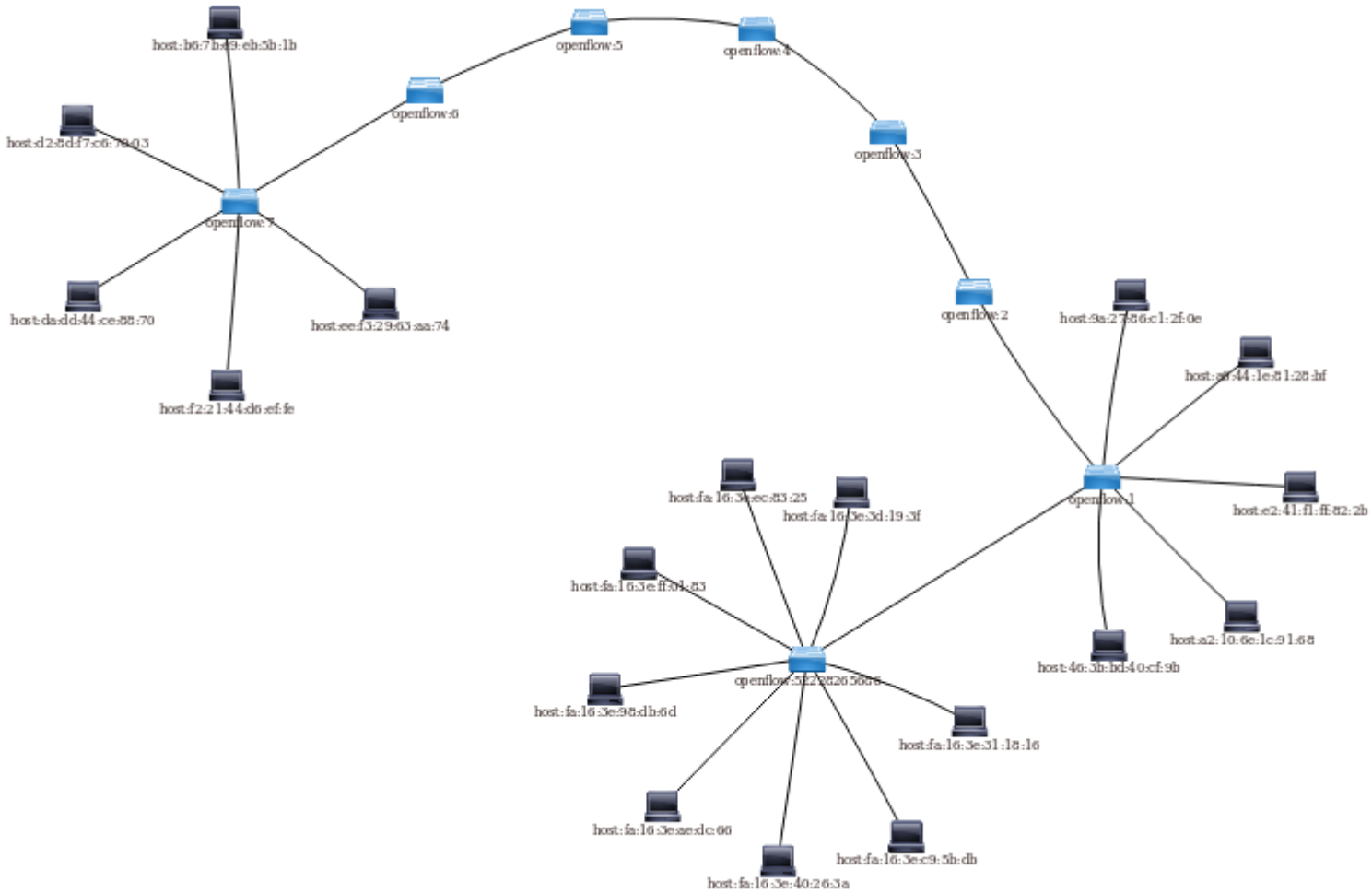

Figura 44 Integración entre Opendaylight y Openstack

Pruebas de conexión entre Openstack con Opendaylight.

Para la prueba de conexión entre una instancia que tenemos creada en el Openstack, se tomó la Instancia Aplicación con la dirección IP 10.1.1.8. Como se muestra en la Figura 45.

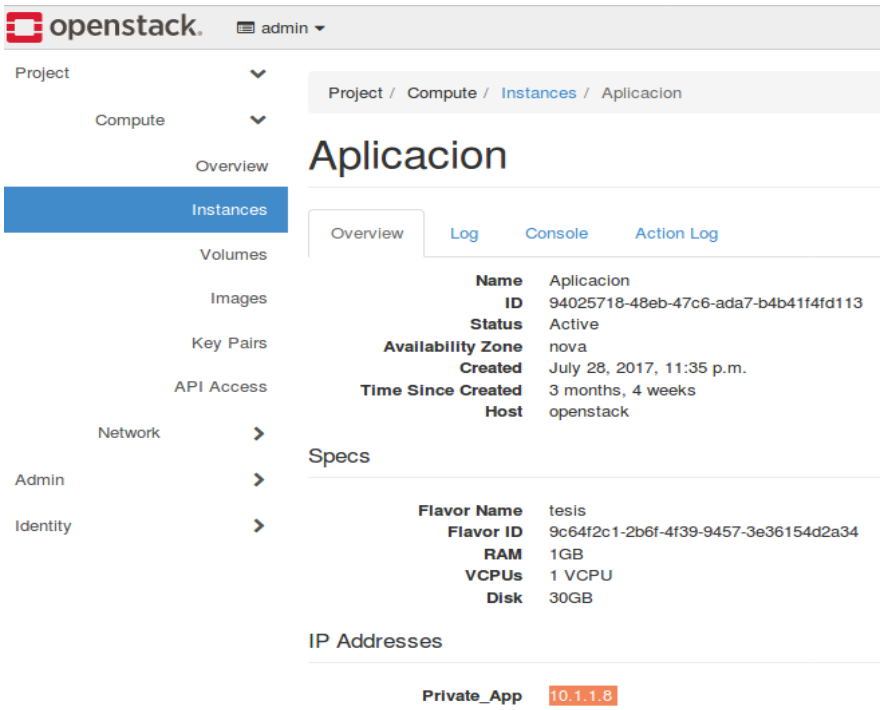

Figura 45 Dirección IP de Instancia de Openstack

Para la prueba se tomó una maquina (host) creada en la topología y con el comando “ping”, se realizó la prueba de conectividad, con la herramienta wireshark se puede visualizar la favorable respuesta que tenemos entre la Instancia y maquina respectiva. Como se muestra la Figura 46.

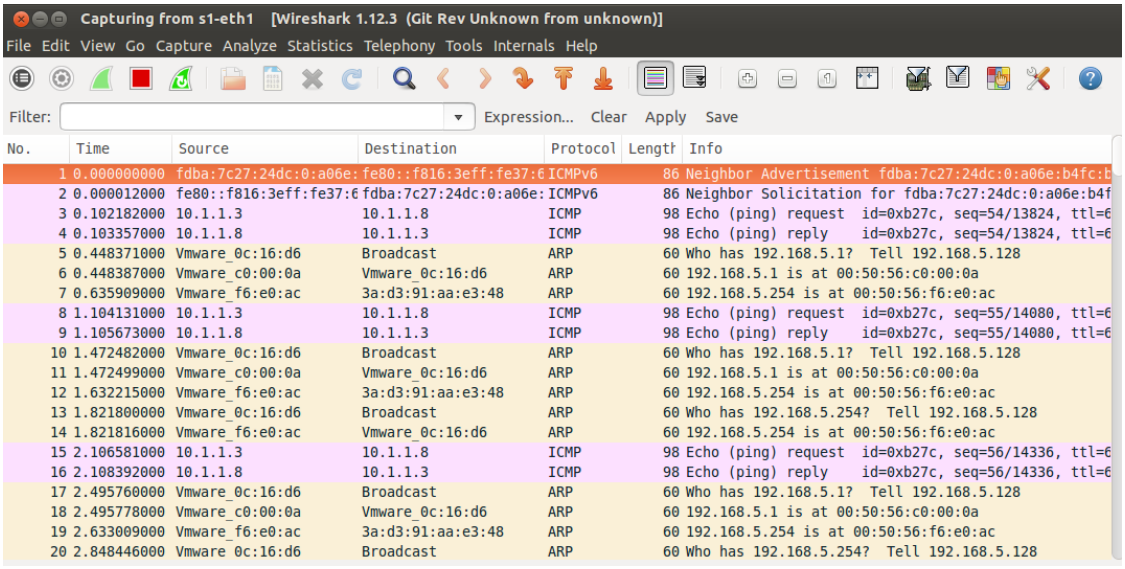

Figura 46 Prueba entre la Instancia y maquina (host)

Prueba de seguridad

Para las pruebas de seguridad se procedió al bloqueo de todos los puertos y se habilito solo los puertos que se necesitan.

La plataforma de Openstack permite cerrar todos los puertos y habilitar solo los puertos que se necesitan como el 22, 8009, 8080 y la base de datos Mysql 3306. Por consiguiente se da seguridad en cada una de las instancias creadas. Como podemos observar en la Figura 47.

☐	Ingress	IPv4	TCP	22 (SSH)	0.0.0.0/0	-	Delete Rule
☐	Ingress	IPv4	TCP	80 (HTTP)	0.0.0.0/0	-	Delete Rule
☐	Ingress	IPv4	TCP	443 (HTTPS)	0.0.0.0/0	-	Delete Rule
☐	Ingress	IPv4	TCP	3306 (MYSQL)	0.0.0.0/0	-	Delete Rule
☐	Egress	IPv4	TCP	8009	0.0.0.0/0	-	Delete Rule
☐	Ingress	IPv4	TCP	8009	0.0.0.0/0	-	Delete Rule

Figura 47 Firewall de Openstack puertos abiertos

Análisis de puertos con Nmap

Mediante la herramienta de nmap, que es un programa de código abierto que sirve para poder administrar y auditar la seguridad de redes en los ordenadores, permite analizar los puertos que tenemos nuestro sistema.

El comando para poder analizar el escaneo de puerto en el protocolo IPv6, es el siguiente:

- nmap el nombre del hosts o la dirección

Como podemos ver en la Figura 48. Hemos analizado los puertos de cada una de nuestras instancias. Dentro de las instancias hemos configurado de la misma forma, cerrando todos los puertos y habilitando solo los que necesitamos.

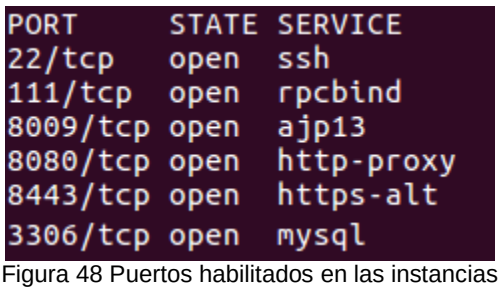

Figura 48 Puertos habilitados en las instancias

Para la comprobación hemos realizado la prueba con el protocolo ICMP, que está bloqueado. Se realizó la prueba con el ping hacia las instancias que están creadas, los resultados fueron que no teníamos comunicación hacia las instancias. Como se muestra en la Figura 49.

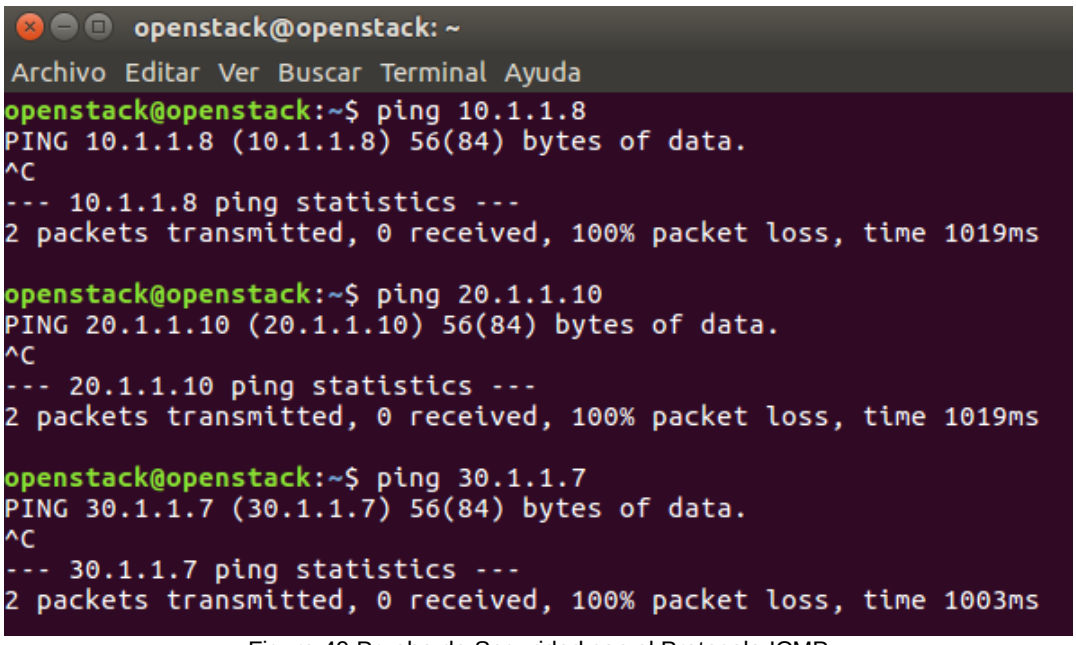

Figura 49 Prueba de Seguridad con el Protocolo ICMP

Balanceo de carga

El balanceo de carga en este proyecto se realizó en el controlador de Opendaylight(ODL), con la configuración de un clúster utilizando un script en el ODL llamado configure_cluster.sh.

El balanceo de carga se implementa por la cantidad de usuarios concurrentes y el crecimiento de la red, por consiguiente el tráfico de red se reparte entre los diversos controladores. Además, la configuración de los controladores permite tener un nodo maestro y varios esclavos, si falla el nodo maestro automáticamente un esclavo toma el control sin que la red pierda conectividad y por consiguiente se evita la perdida de

información que circula por la red.

Script configure_cluster.sh

Este script tiene definido un mecanismo que permite que múltiples procesos y programas trabajen juntos como una sola entidad. Teniendo como principales ventajas la alta disponibilidad, en el cual si un nodo falla el otro nodo sigue funcionando y disponible para la red; además proporciona persistencia es decir que no se perderá ningún dato de la información en Opendaylight.

El mecanismo realiza un identificador único para cada instancia, la realización de este identificador a cada nodo se le asigna un numero de role, esta función se define en el archivo akka.conf en el parámetro member-1.

Así mismo en cada nodo se configura un módulo, en este caso de manera predeterminada todos los datos se colocan en un fragmento en la cual se configura las réplicas en el archivo module-shards.conf, de acuerdo a los nodos que se vaya a utilizar.

Configuración en cada controlador Opendaylight

Se tiene un clúster de dos nodos controladores, cada miembro tiene un identificador.

bin/configure_cluster.sh <index> <seed_nodes_list>

- índice(index): es un número entero de 1..N, donde N es el número de nodos. Esto indica qué nodo controlador está configurado por el script como el maestro.
- Lista de nodos(seed_nodes_list): lista de nodos de la manera que se va a inicializar, se escribe a dirección IP de forma separada por coma o espacio.

Configuración del primer nodo “Maestro: 192.168.1.111”. Tal como la Figura 50.

- ./bin/configure_cluster.sh 1 192.168.1.111 192.168.1.112

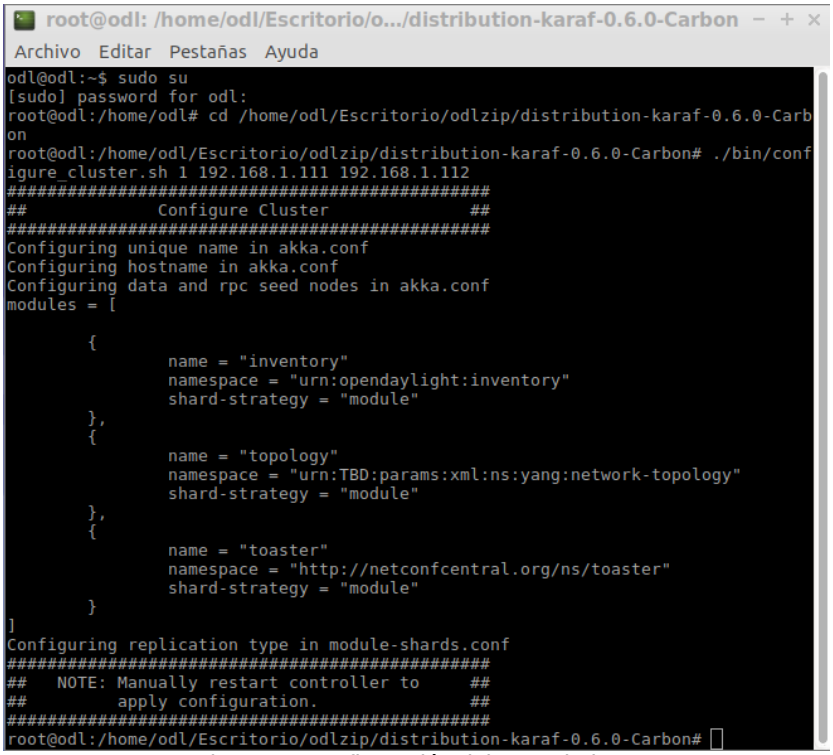

Figura 50 Configuración del controlador 1

Configuración del segundo nodo “Esclavo: 192.168.1.112”. Tal como la Figura 51.

- ./bin/configure_cluster.sh 2 192.168.1.111 192.168.1.112

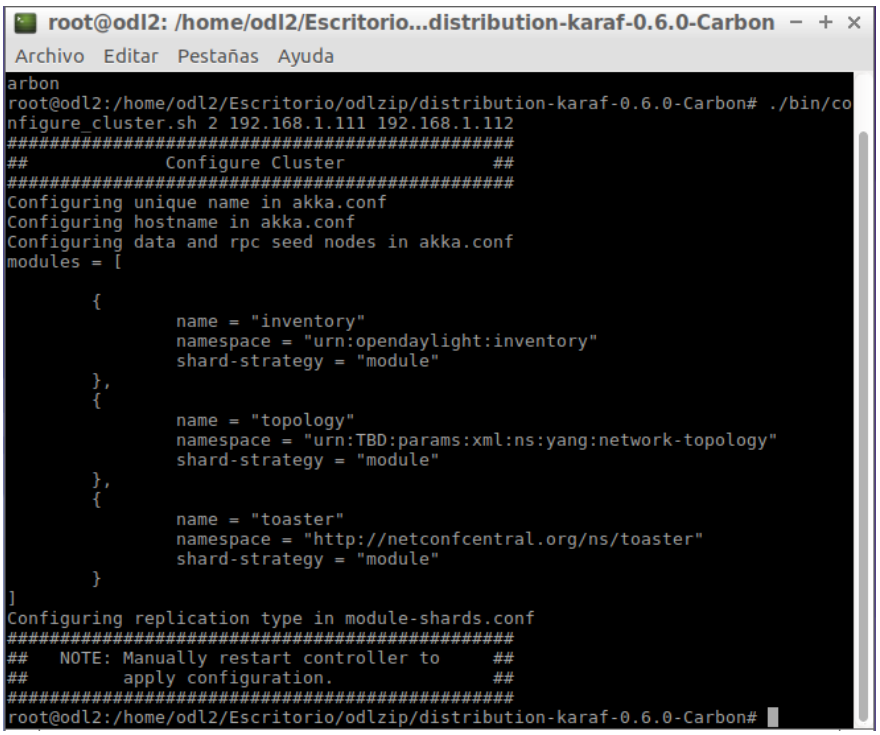

Figura 51 Configuración del controlador 2

Pruebas de balanceo de carga

Para el balanceo de carga la prueba se realizó tomando dos controladores, estos funcionando al mismo tiempo. La configuración realizada para este proyecto el nodo maestro es la dirección 192.168.1.111 y el esclavo es la 192.168.1.112.

En la siguiente Figura 52, se puede observar que los dos controladores están funcionando correctamente y teniendo comunicación con el simulador de Mininet.

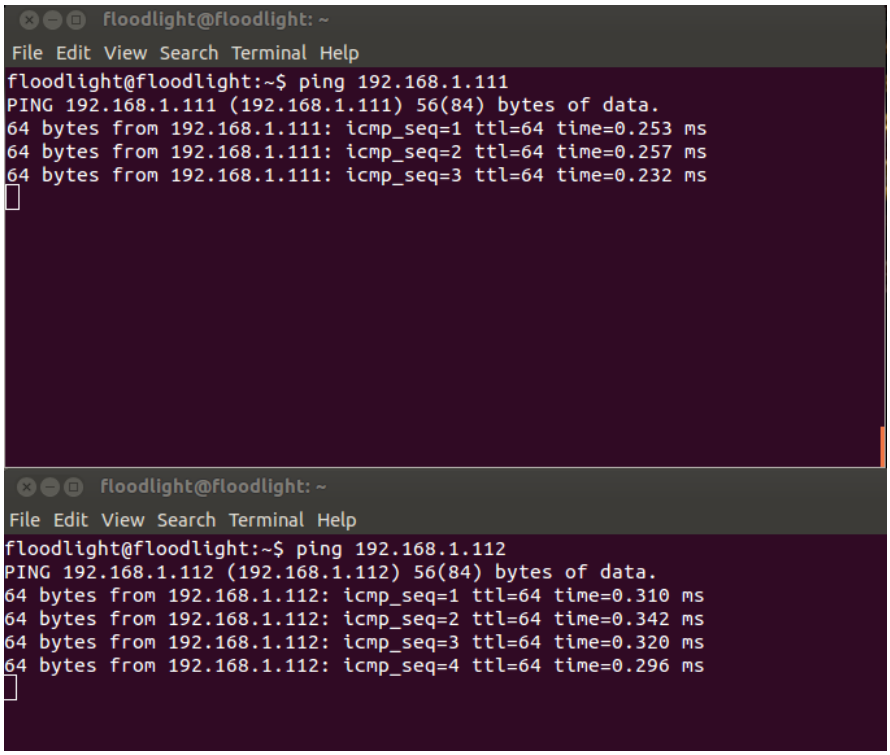

Figura 52 Conectividad entre Mininet y los dos Controladores

Luego procedemos a realizar un ping entre dos host creados mediante el script ejecutado en mininet, posteriormente con la herramienta Wireshark se puede observar como es la comunicación entre los dos nodos.

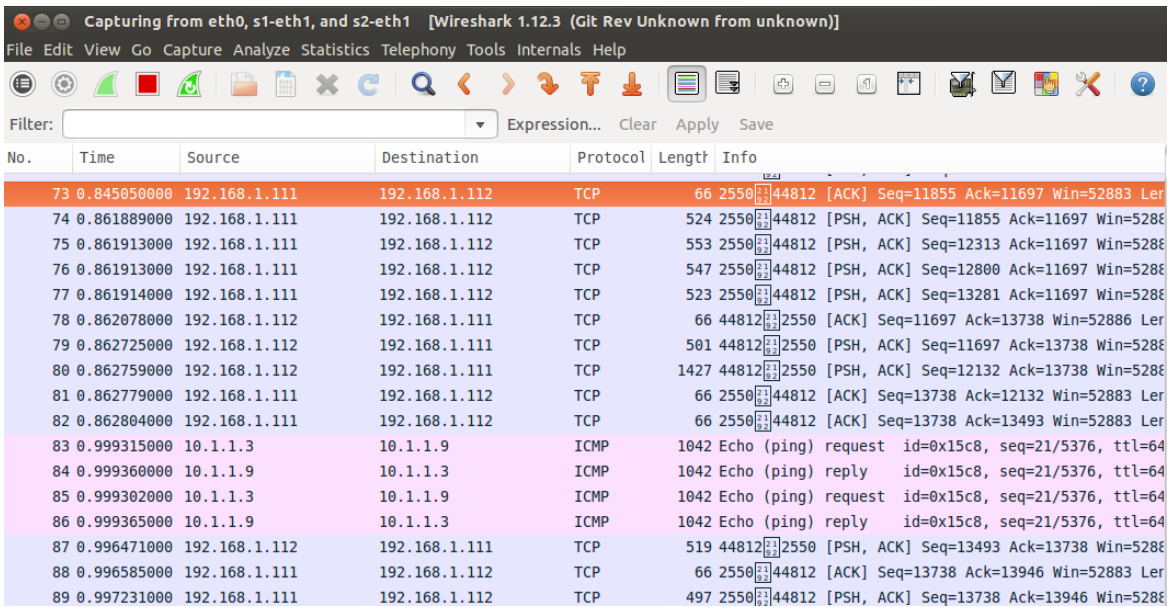

Figura 53 Verificación del balanceo entre los controladores

Como se observa en la Figura 53, la comunicación se realiza correctamente entre los controladores, lo que se puede visualizar es el balanceo de cargo entre los hosts y en la retransmisión entre los dos controladores; y los hosts tienen conectividad correcta.

Ahora procedemos a deshabilitar al controlador maestro, para realizar la prueba de alta disponibilidad. Como se muestra en la siguiente Figura 54.

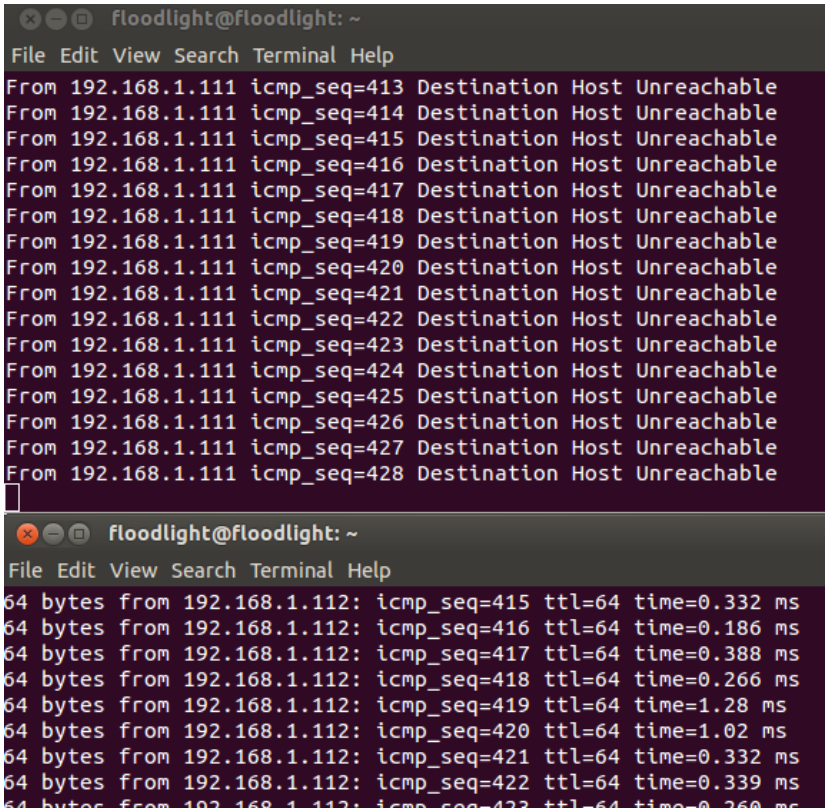

Figura 54 Deshabilitación del Controlador Maestro

Se puede observar que ya está deshabilitado el nodo maestro 192.168.1.11 y se tiene solo conexión con el nodo esclavo 192.168.1.112.

Se procede a realizar la prueba por mediante wireshark, como se procedió a deshabilitarlo se va perdiendo conexión entre el un controlador con el mininet, se observa cómo se va retransmitiendo entre los dos controladores y también se visualiza que no se ha perdido conexión entre los dos hosts. Como se muestra en la Figura 55.

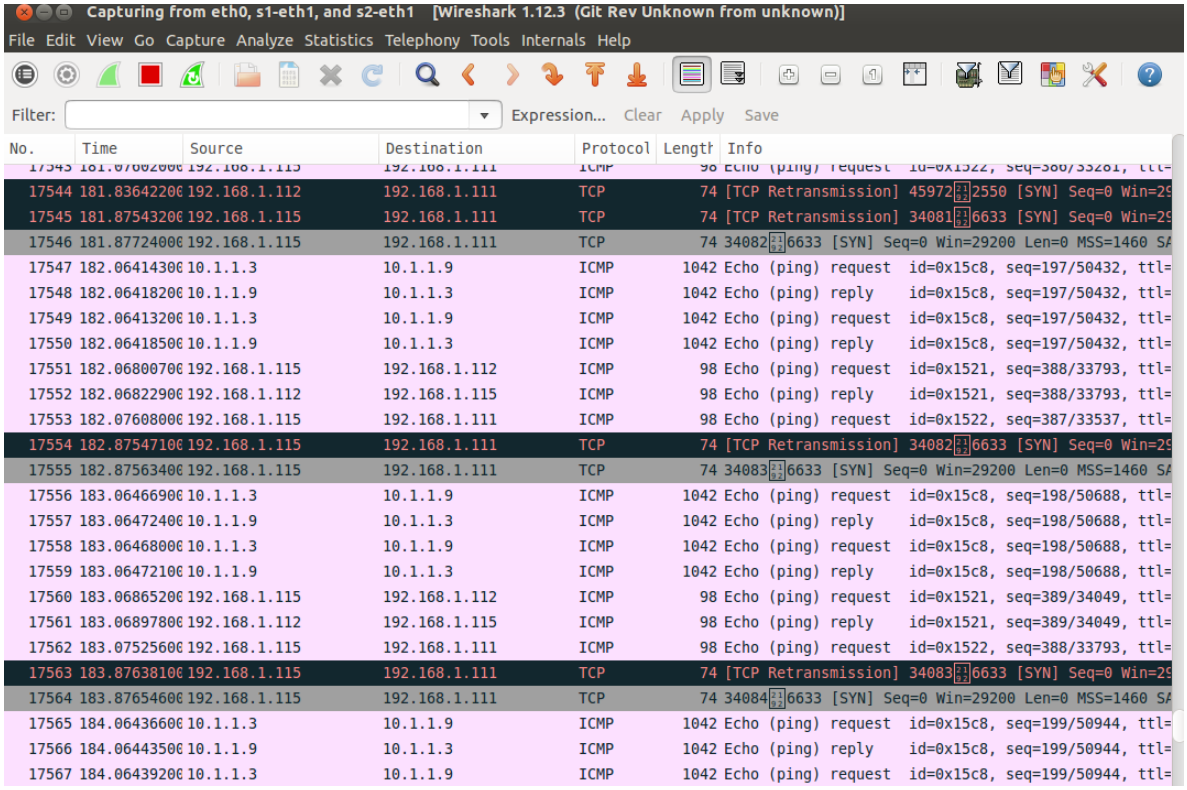

Figura 55 Verificación del balanceo de carga

Podemos ver también con el ping entre los dos hosts, no se perdió luego de haber deshabilitado el nodo maestro, es así que el nodo esclavo toma el control.

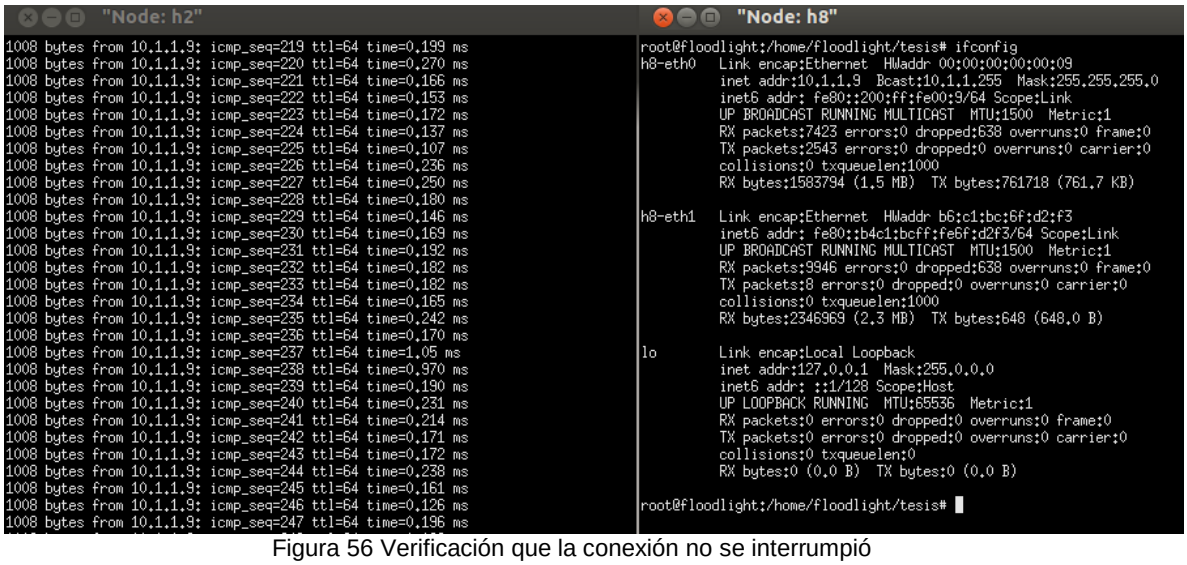

Figura 56 Verificación que la conexión no se interrumpió

Análisis de Resultados:

Luego de realizar el estudio bibliográfico y el análisis del estudio del arte se procedió a realizar diversos escenarios que validen nuestra hipótesis de usar las redes SDN como sistema de transición de IPv4 a IPv6.

En el primer escenario se tuvo la dificultad en el software de Mininet, si bien permite de manera nativa los flujos en IPv4 e IPv6, estos flujos no nos permiten trabajar con los mecanismos de transición como el túnel GRE y NAT; esto se debe a que configuraciones internas propias de la máquina virtual.

De igual manera en el segundo escenario se tuvo el mismo inconveniente, sin embargo se hice las pruebas con dualstack y las pruebas fueron afirmativas, lo que se puede concluir que mininet soporta y si tuviéramos una arquitectura parecida a la que está siendo emulada, se podría aplicar SDN para lo que es como sistema de transición es dualstack.

En el tercer escenario se realizó la interconexión entre dos hosts de diferentes redes, se utilizó con otro controlador llamado pox, que agrega automáticamente los gateways se tenía la conexión entre las diferentes redes y a su vez permite la configuración automática de los flujos.

En el cuarto escenario se utilizó el controlador Opendaylight con la implementación del mecanismo dualstack, que es el único mecanismo que nos permite trabajar mininet y las pruebas fueron satisfactorias.

Es necesario mencionar que se analizaron los distintos puntos de fallo, tanto en el controlador, openflow y el sistema de reenvío en Mininet, en mininet nos encontramos con el error que este no permitía hacer el cambio de flujo de forma automática, debido que la versión actual de mininet no cuenta con esa función.

Si bien es cierto que las redes SDN soportan el protocolo IPv6, no están listas para trabajar en integramente en IPv6 y en los mecanismos de transición, de acuerdo al artículo [25] el único mecanismo que se puede implementar en la redes actuales SDN es dualstack.

El uso de dispositivos como JUNIPER, HP, Cisco entre otros dispositivos comerciales soportan los mecanismos de transición como tunneling, nat y dual Stack, lamentablemente debido a las limitaciones de presupuesto no se pudo realizar pruebas utilizando estos dispositivos. Sin embargo en el artículo "*Mecanismos de Transición hacia redes IPv6*" [23] se realizó los mecanismo de transición en Cisco, como también en el documento "Diseño e implementación de una red IPv6 para transición eficiente desde

IPv4” [24] se implementó en Juniper.

El controlador es uno de los puntos más débiles dentro de una red SDN, por lo cual es necesario que el controlador tenga alta redundancia y alta disponibilidad ya que de este dispositivo depende en gran medida el flujo de la información a través de una red SDN. En nuestra propuesta se utilizó dos controladores OpendayLight, en una configuración de Cluster para balanceo de carga y redundancia; con lo cual si un controlador queda fuera de servicio el segundo controlador permanece activo sin que la red SDN pierda conectividad.

Conclusiones:

Las redes definidas por software (SDN) centralizan las tablas de flujos de datos, simplificando el despliegue y la administración de la red, mejorando notablemente el tiempo de trabajo del administrador de red.

Las funciones de red virtualizadas (NFV) reducen los dispositivos de hardware en gran medida, ya que por medio de las funciones de red los routers, switches y servidores ahora van a ser virtuales, lo que ocasiona también es reducir los costos de adquisición de equipos.

Las empresas en sus nuevos proyectos de red están combinado NFV y SDN, para tener mayor rentabilidad en las comunicaciones, esto se lleva acabo realizando alianzas entre las empresas de software y cloud computing, generando redes de nueva generación para los avances tecnológicos que se avecinan.

Las tecnologías SDN y NFV nos ayudan en la generación de mayor flexibilidad, escalabilidad, compatibilidad e integración entre las diferentes arquitecturas de redes, gestionando de una forma ágil la migración, movilidad y la interoperabilidad entre redes.

Los mecanismos de transición de IPv4 a IPv6, por el momento se va implementando paulatinamente en las empresas, siendo el mecanismo más utilizado por los administradores de red es DualStack, que no es el más eficiente sin embargo minimiza el impacto de la migración y permite regresar de forma rápida a la configuración anterior si la transición no es eficiente, reduciendo costos y tiempo de despliegue, la dificultad de este mecanismo es que se debe tener una IPv4

Es posible aplicar la lógica del controlador a una red SDN existente con relativamente pocas modificaciones en el código, para ayudar a proveedores de servicios a llevar a cabo un balanceo de carga entre sus equipos, adecuándolos al tipo de aplicación.

La seguridad distribuida reduce la superficie de ataque mejorando el control, el monitoreo y gestión de la red, puntualizando los puntos de fallo de la red. A esto se le deben sumar políticas dentro de otras infraestructuras hasta llegar al cliente generando perímetros de seguridad en diferentes grados, de esta forma se limita al atacante dificultándole el control total en la red.

Bibliografía:

[1] http://www.zakon.org/robert/internet/timeline/

[2] T. Peter, H. Pavol and D. Lukáš, "Implementation and evaluation of IPv6 to IPv4 transition mechanisms in network simulator 3," 2016 International Conference on Systems, Signals and Image Processing (IWSSIP), Bratislava, 2016, pp. 1-4.

[3] S. Hagen, "IPv6 Essentials Integrating IPv6 into your IPv4 network" in , Sebastopol, CA:O'Reilly Media, 2014.

[4] S. Rani Vandana, "Analysis of IPv6/IPv4 dual-stack transition mechanism", Int. J. of Science Eng. and Tech. Research, vol. 4, no. 6, pp. 1968-1971, 2015.

[5] P. Grayeli, S. Sarkani, T. Mazzuchi, "Performance analysis of IPv6 transition mechanisms over MPLS", Int. J. of Com. Networks and Inf. Security, vol. 4, no. 2, pp. 91-103, 2012.

[6] P. Wu, Y. Cui, J. Wu, J. Liu, C. Metz, "Transition from IPv4 to IPv6: A state-of-the-art survey", IEEE Commun. Surveys Tuts., vol. 15, no. 3, pp. 1407-1424, 2013.

[7] G. Shang, P. Zhe, X. Bin, H. Aiqun and R. Kui, "FloodDefender: Protecting data and control plane resources under SDN-aimed DoS attacks," IEEE INFOCOM 2017 - IEEE Conference on Computer Communications, Atlanta, GA, 2017, pp. 1-9.

[8] M. Casado et al., "Ethane: Taking control of the enterprise", ACM SIGCOMM Comput. Commun. Rev., vol. 37, no. 4, pp. 1-12, 2007.

[9] T. Huang, F. R. Yu, C. Zhang, J. Liu, J. Zhang and Y. Liu, "A Survey on Large-Scale Software Defined Networking (SDN) Testbeds: Approaches and Challenges," in IEEE Communications Surveys & Tutorials, vol. 19, no. 2, pp. 891-917, Secondquarter 2017.

[10] R. Chayapathi; S. Farrukh Hassan; Paresh Shah “Network Functions Virtualization (NFV) with a Touch of SDN” Cover image for Network Functions Virtualization (NFV) with a Touch of SDN publisher logo Network Published by Addison-Wesley Professional, 2016.

[11] “Open Networking Foundation,” https://sdndirectory.opennetworking.org/.

[12] S. Evans, "ComputerWeekly.com - The history of OpenFlow," http://www.computerweekly.com/feature/The-history-of-OpenFlow.

[13]https://www.os3.nl/_media/20132014/courses/ssn/projects/practical_security_analysis_of_openflow_report.pdf

[14] B. H. N. H. B. J. V. Lantz, 2015 septiembre 2015. [En línea]. Available: https://github.com/mininet/mininet/wiki/Introduction-to-Mininet.

[15]Juan, «Ubuntu guia,» 28 Abril 2014. [En línea]. Available: http://www.ubuntu-guia.com/2012/04/instalar-oracle-java-7-en-ubuntu-1204.html. [Último acceso: 2017 julio 24].

[16]S. Mike, «Serenity Networks,» 28 02 2017. [En línea]. Available: https://serenity-networks.com/how-to-install-openstack-ocata-on-a-single-server-using-devstack/. [Último acceso: 10 04 2017].

[17]«mininet.org,» [En línea]. Available: mininet.org/overview/. [Último acceso: 2017].

[18]Mininet, «Mininet website,» [En línea]. Available:

https://github.com/mininet/mininet/wiki/Introduction-to-Mininet#how.

[19]L. Mark, «sdxCentral,» 2017. [En línea]. Available: https://www.sdxcentral.com/articles/contributed/dimensioning-openstack-neutron-nfv-systems-mark-lambe/2014/09/. [Último acceso: 15 08 2017].

[20]H. J. Wang, «A personal Comunication Service Creation Model,» IEEE.

[21]B. H. N. H. B. J. V. Lantz, 2015 septiembre 2015. [En línea]. Available: https://github.com/mininet/mininet/wiki/Introduction-to-Mininet.

[22]S. Braun, «Yet technologies,» [En línea]. Available: http://www.yet.org/2014/09/openvswitch-troubleshooting/.

[23] Taffernaberry, J. C. (2011). Mecanismos de Transición hacia redes IPv6 (Doctoral dissertation, Facultad de Informática).

[24] B. F. G. O. P. Luis, 12 noviembre 2012. [En línea]. Available: http://www.scielo.org.co/pdf/inco/v14n2/v14n2a17.pdf.

[25] Ciprian, «techtarget.com,» 2015. [En línea]. Available: http://searchsdn.techtarget.com/feature/IPv6-SDN-When-worlds-collide-in-a-good-way

Anexos

Anexo1

Pasos previos a la instalación de los Programas

Antes de la instalación de Openstack y de Opendaylight, necesitamos lo siguiente:

Actualizar Repositorios

Digitar los siguientes comandos en la consola de Linux:

- sudo apt-get update
- sudo apt-get upgrade

Instalar Git

Lo necesitamos para poder bajar las librerías y paquetes para la instalación de los programas.

- sudo apt-get install git*

Instalar Java JDK

Para la instalación necesitamos realizar el siguiente procedimiento:

Añadir repositorio

- sudo add-apt-repository ppa:webupd8team/java

Actualizar Repositorios

Para adicionar correctamente el repositorio agregado.

- sudo apt-get update

Instalar Java de Oracle

- sudo apt-get install oracle-java8-installer

Establecer Variables de Entorno

- sudo update-java-alternatives -s java-8-oracle

Anexo 2

Instalación de OpenDayLight (ODL)

La instalación de ODL se realizó sobre el sistema operativo Linux, utilizamos la distribución Lubuntu de 64 bits en una ambiente virtual, con una memoria RAM de 4 GB.

https://help.ubuntu.com/community/Lubuntu/GetLubuntu/LTS

Descargar OpenDayLight

En la página oficial de OpenDayLight, descargar la distribución estable en este caso **Carbon.**

- https://www.opendaylight.org/downloads

Ejecutar OpenDayLight

a) Descomprimir el archivo.
   - unzip distribution-karaf-0.6.1-Carbon.zip

b) Entrar a la carpeta de opendaylight
   - cd distribution-karaf-0.6.x-Carbon

c) Ejecutamos el archivo karaf.
   - ./bin/karaf

Se empieza a ejecutar el controlador y una vez terminado nos presenta lo siguiente en la consola, como se observa en la Figura 57.

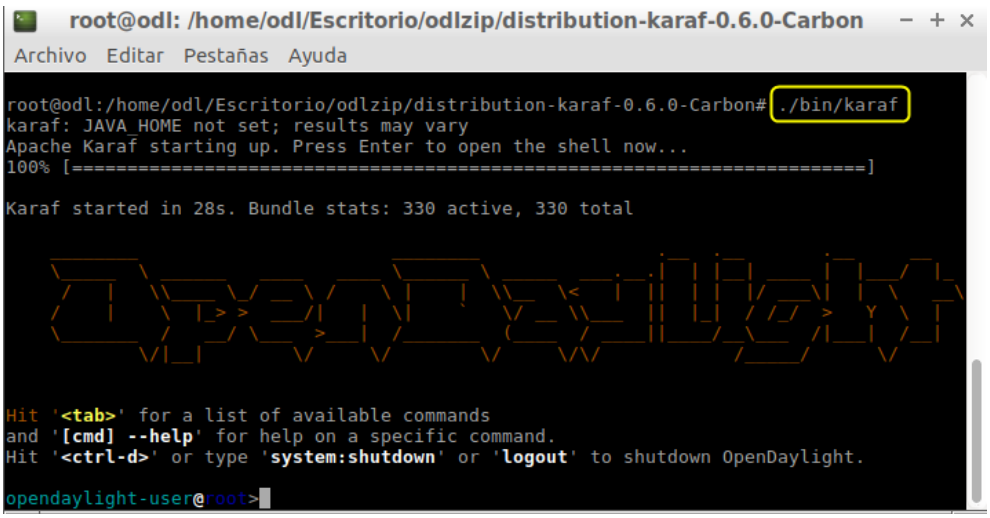

Figura 57 Consola de OpenDayLight

Instalación de las características en ODL

Las características que se instalaran, son módulos de configuración de la red de datos desarrollados por ODL.

Los módulos necesarios se instalaran de la siguiente manera:

- feature:install odl-dlux-core odl-restconf-all odl-mdsal-apidocs odl-dluxapps-topology odl-dluxapps-nodes odl-dluxapps-yangvisualizer odl-l2switch-all odl-l2switch-switch-ui http odl-mdsal-clustering odl-openflowplugin-flow-services

Al ser instalados los módulos, en nuestro navegador favorito se debe ingresar a la dirección:

- http://localhost:8181/index.html

Se puede visualizar la siguiente pantalla de ingreso, aquí procedemos a ingresar las credenciales por defecto de Opendaylight, estas son:

- username: admin
- Password: admin.

Tal como se visualiza en la Figura 58.

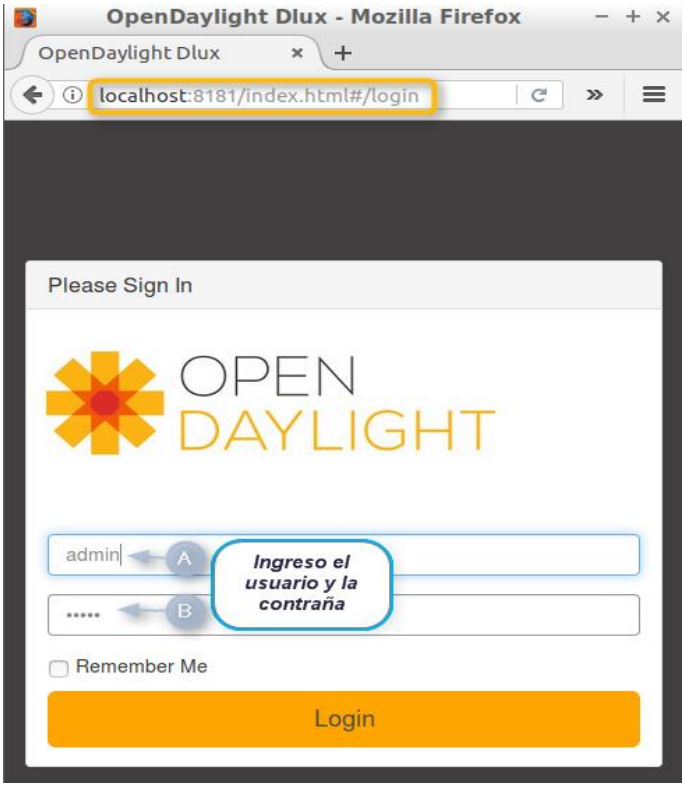

Figura 58 Pantalla de Inicio de Sesión de OpenDayLight

Ya ingresadas las credenciales correctamente, se puede observar en la figura 59 la pantalla principal de ODL.

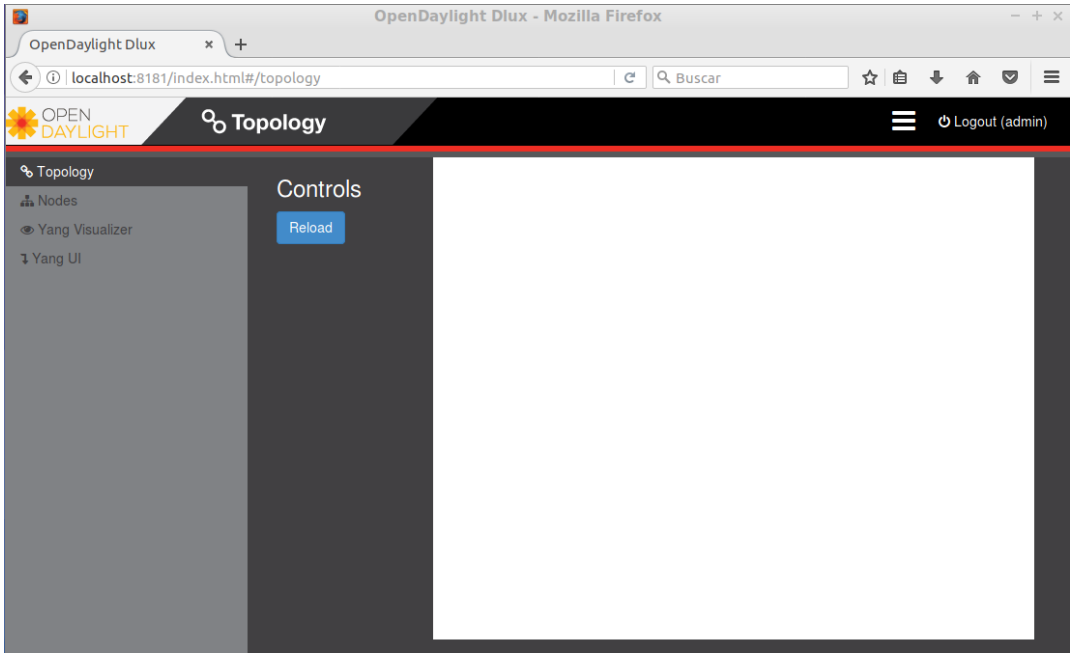

Figura 59 Pantalla Principal de OpenDayLight

Anexo 3

Emulador Mininet

Para la emulación de un sistema de red se utilizara el emulador de Mininet, este software nos permite implementar redes virtuales y con el controlador OpenDayLight.

1. **Descargar Mininet**
   Descargar la máquina virtual preinstalada.
   - http://opennetlinux.org/binaries/floodlight-vm.zip
2. **Iniciar la Máquina**
   La ejecución se realizara en Vmware Workstation (programa para virtualización), en el cual será iniciada con el sistema operativo que es Ubuntu 14 y con los parámetros establecidos como: 1 Gb de RAM y un procesador de un núcleo.

Luego de haber creado la máquina, se procede a iniciarla. Por lo tanto como es una maquina preinstalada Mininiet, para ingresar a la maquina procedemos a ingresar el usuario y la contraseña:

- Usuario: floodlight
- Contraseña: floodlight

Creación de topologías de red en mininet

Ya dentro de la plataforma de mininet, se procederá a probar el concepto citado de SDN, que es la separación del plano de control y de datos en un controlador.

Para lo cual, se va a ir mostrando la utilidad del comando “mn” mediante sudo mn –h:

En la interfaz de línea de comando CLI de la máquina virtual Mininet desplegaremos los comandos iniciales como se muestra en la figura 60.

```
floodlight@floodlight:~$ sudo su
[sudo] password for floodlight:
Sorry, try again.
[sudo] password for floodlight:
root@floodlight:/home/floodlight# sudo mn -h
Usage: mn [options]
(type mn -h for details)

The mn utility creates Mininet network from the command line. It can create
parametrized topologies, invoke the Mininet CLI, and run tests.

Options:
  -h, --help            show this help message and exit
  --switch=SWITCH       default|ivs|lxbr|ovs|ovsbr|ovsk|user[,param=value...]
                        ovs=OVSSwitch default=OVSSwitch ovsk=OVSSwitch
                        lxbr=LinuxBridge user=UserSwitch ivs=IVSSwitch
                        ovsbr=OVSBridge
  --host=HOST           cfs|proc|rt[,param=value...]
                        rt=CPULimitedHost{'sched': 'rt'} proc=Host
                        cfs=CPULimitedHost{'sched': 'cfs'}
  --controller=CONTROLLER
                        default|none|nox|ovsc|ref|remote|ryu[,param=value...]
                        ovsc=OVSController none=NullController
                        remote=RemoteController default=DefaultController
                        nox=NOX ryu=Ryu ref=Controller
  --link=LINK           default|ovs|tc[,param=value...] default=Link
                        ovs=OVSLink tc=TCLink
  --topo=TOPO           linear|minimal|reversed|single|torus|tree[,param=value
                        ...] linear=LinearTopo torus=TorusTopo tree=TreeTopo
                        single=SingleSwitchTopo
                        reversed=SingleSwitchReversedTopo minimal=MinimalTopo
```

Figura 60 Utilidad sudo mn –h

De la figura anterior, se puede destacar los siguientes parámetros:

--switch.- Por defecto trabaja con Open vSwitch, se implementa y funciona como un conmutador normal.

--controller.- Por defecto trabaja con un controlador OpenFlow y que está preinstalado en la máquina virtual. Con nox y ryu se trabaja con los controladores del mismo nombre y si no se posee un controlador se mantiene a la escucha de algún controlador remoto.

--topo.- Se crea una topología de red virtual, teniendo diferentes opciones: minimal crea una topología por defecto de un conmutador y dos hosts; single conectado a un controlador determinado a “X” número de hosts; linear crea “X” conmutadores conectados uno detrás de otro, cada uno con un host conectado.

--host.- se crean host de forma automática con una dirección IP.

Creación de una topología de red mediante sudo mn sin controlador

Para realizar una red sencilla, se puede realizar escribiendo directamente el siguiente comando:

sudo mn - -topo=single, 4

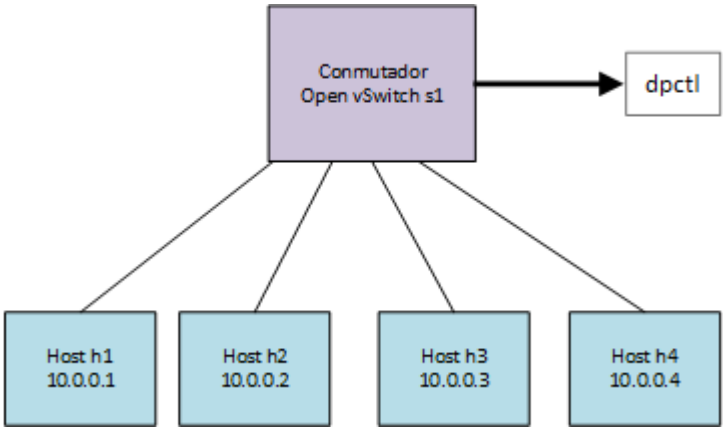

Figura 61 Topología de red mediante sudo mn

La figura 61 demuestra que sea crea una topología que no está utilizando un controlador, por consiguiente el conmutador no puede realizar el envío de flujos, por lo que se utilizara dpctl, que es una herramienta de configuración que permite el acceso directo y de manera visible una tabla de flujo.

Creación de una topología por script con un controlador remoto

Se plantea la creación de topologías en Mininet, en el cual se integra con un controlador externo que es Opendaylight (ODL), para la visualización de la red creada y la información que está en los nodos.

Mediante Scripts en Python

En primer lugar se creara una topología de red, ésta se realiza mediante un script programado en el lenguaje de Python que interactúa con Mininet, la extensión del script debe ser .py. En este caso no se profundizara en el significado de cada línea de código, lo que se explicara es los paquetes necesarios para mininet, el cómo añadir cada componente y como se ejecuta posteriormente desde Mininet.

Creación de Scripts para Mininet

Para la creación de un script en python que interactúe con mininet, se debe importar los paquetes o clases necesarias para la conexión con Mininet, los cuales son:

> ***from mininet.net import Mininet:*** clase principal para crear y administrar una red
>
> ***from mininet.node import Controller,RemoteController:*** realiza la creación de la conexión con el controlador ODL.

from mininet.cli import CLI: proporciona la interfaz de línea de comandos que es invocada por la red y nos proporciona comandos útiles para una configuración.

from mininet.topo import Topo: clase que realiza la creación de la topología de red.

Parámetros para la creación de la red

Para la creación de la red se debe tener en cuenta los siguientes métodos:

net.addHost: agrega un host a una topología y devuelve el nombre de host.

net.addSwitch: agrega un switch a una topología y devuelve el nombre del switch.

net.addController: agrega un controlador a la topología.

self.addLink: agrega un enlace bidireccional a una topología.

Requerimientos y desarrollo del script

El script se desarrolla teniendo un esquema de requerimientos de red; cuantos hosts, switches, controladores se va a utilizar en la infraestructura. De acuerdo a esto se procederá a programar el script que se va desplegar en Mininet.

La programación del script se establece en el siguiente orden:

- Importar los paquetes y clases principales.
- Definir el nombre de la clase topo.
- Agregar los host de acuerdo a los requerimientos de red.
- Agregar los switches y los controladores.
- Crear los enlaces ente el host y el switch.
- Iniciar los controladores.
- Iniciar los switches con cada controlador agregado.
- Se guarda el script con extensión de python que es .py

Posteriormente de haber desarrollado el script:

- Se crea una carpeta para almacenar los scripts, se guardaran los scripts que se desarrollen en la siguiente dirección:

/home/floodlight/mininet/tesis

Para la ejecución del script creado se debe ubicar en la carpeta donde están guardados los scripts y mediante el comando “ls” se observaran los scripts creados, en la Figura 62.

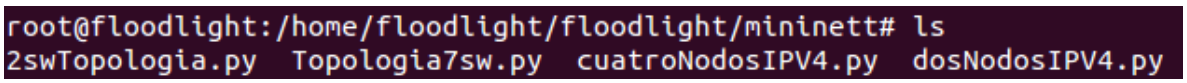

Figura 62 Dirección de almacenamiento de los scripts

Una vez elegido el script se debe ejecutar lo siguiente:

```
sudo python 2swTopologia.py
```

Luego de haber ejecutado, se observa en la Figura 63 el despliegue de los componentes, la creación de los nodos, de los enlaces, el inicio de la red y los switches.

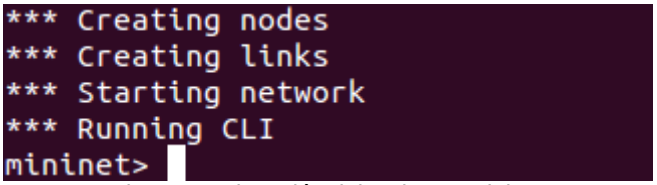

Figura 63 Ejecución del script en mininet

Para comprobar la topología de red creada con sus respectivos hosts y switches, si no hubo ningún error en la interfaz gráfica del controlador, se visualizara en la Figura 64.

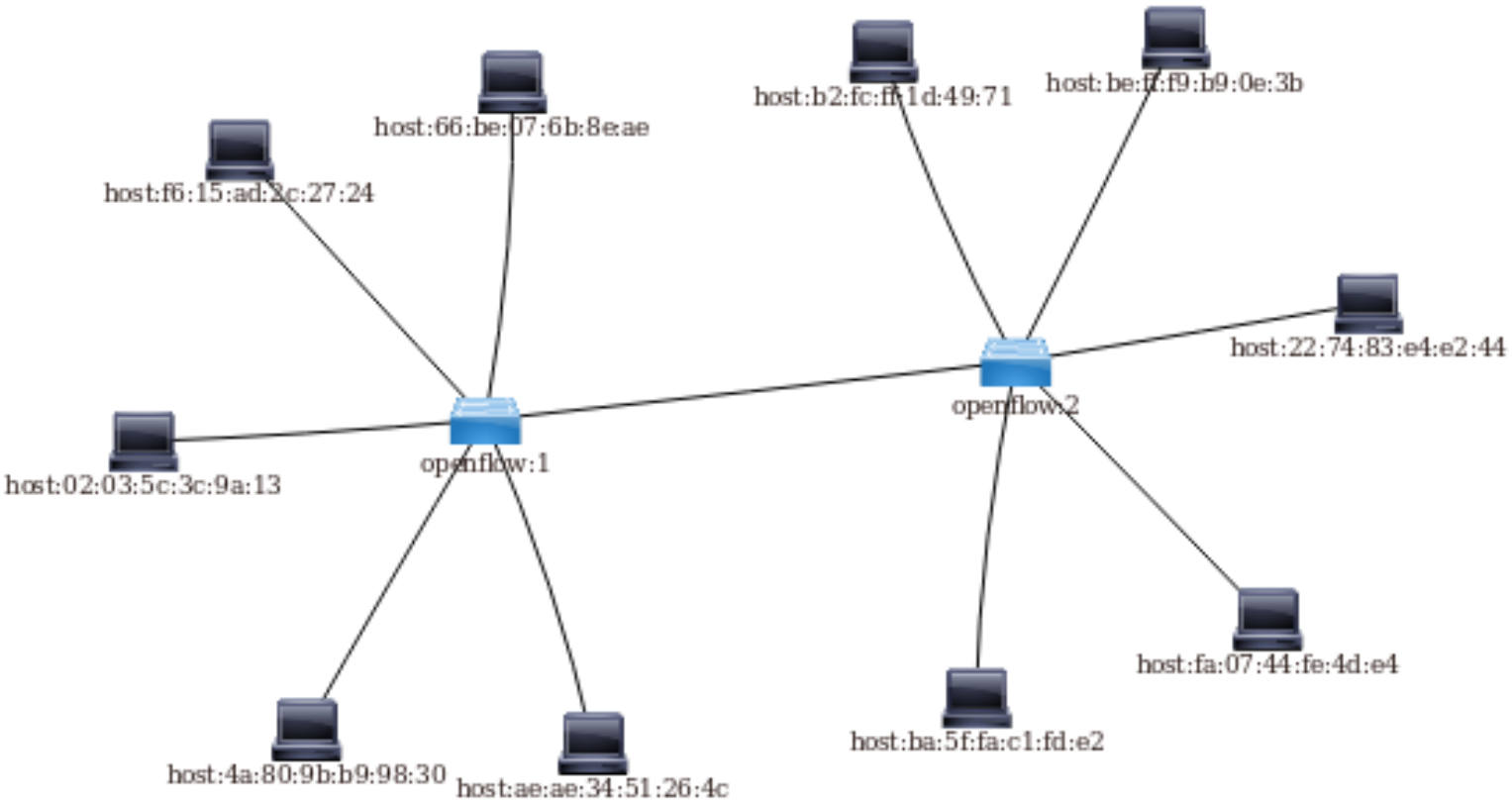

Figura 64 Topologia de la red

Anexo 4

Instalación OpenStack

Para la instalación de OpenStack, debemos tener la versión estable de Linux, vamos a utilizar **Ubuntu 16.04.2 Desktop** [25].

1. Utilizar **git** para clonar **devstack.** Devstack es el encargado de realizar toda la instalación de OpenStack.
    - cd /
    - sudo git clone https://git.openstack.org/openstack-dev/devstack -b stable/ocata

    En nuestro caso vamos utilizar la versión estable de Openstack **ocata**, si desea instalar una versión diferente como: kilo o liberty, simplemente cambie "ocata" por la distribución que desee instalar.
2. Necesitamos copiar el archivo **sample.local.conf**.
    - cd devstack/
    - sudo cp samples/local.conf local.conf
3. A continuación, abri el archivo que se copió y establecemos una contraseña, esta servirá para el despliegue automatizado y para ingresar en la interfaz gráfica OpenStack.
    - sudo nano local.conf
4. Dentro del archivo de configuración buscaremos el término **password variables**. Aquí escribiremos la contraseña de **ADMIN_PASSWORD** y cambiaremos los tres parámetros siguientes por **$ADMIN_PASSWORD.** Como se muestra en la Figura 65.

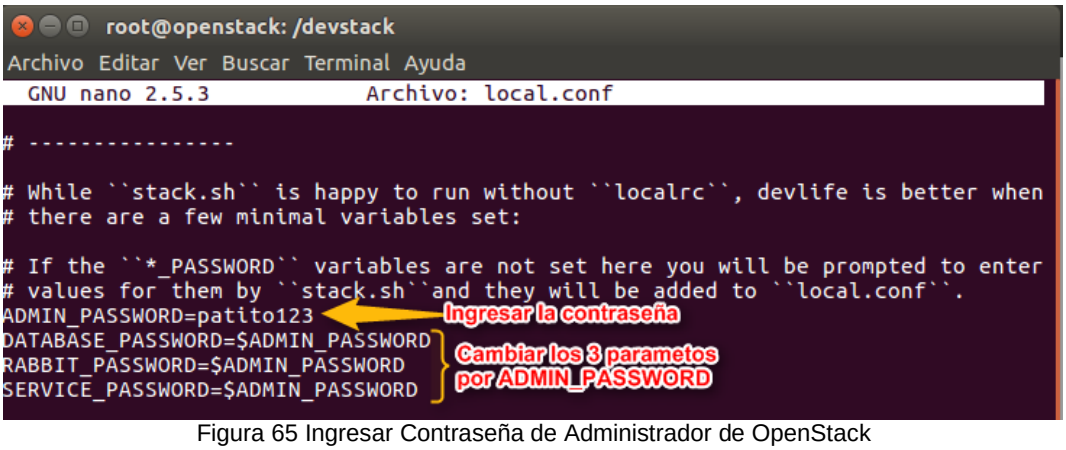

Figura 65 Ingresar Contraseña de Administrador de OpenStack

5. Procedemos a crear un nuevo usuario para OpenStack, a este nuevo usuario lo hacemos el propietario de la carpeta **devstack.**
    - ♦ sudo /devstack/tools/create-stack-user.sh
    - ♦ sudo chown -R stack:stack /devstack

6. Luego de las pre-configuraciones realizadas, se procede a instalar OpenStack. Este proceso se demorara entre 45 minutos a 1 hora, dependiendo de la velocidad de su internet. Digitamos lo siguiente:

    - sudo su stack
    - /devstack/stack.sh

7. Inmediatamente luego de haber terminado de instalar, podremos observar en la siguiente Figura 66, detalladamente la dirección IP por la que debemos ingresar al OpenStack y la contraseña anteriormente configurada con la que accederemos a la interfaz de administración.

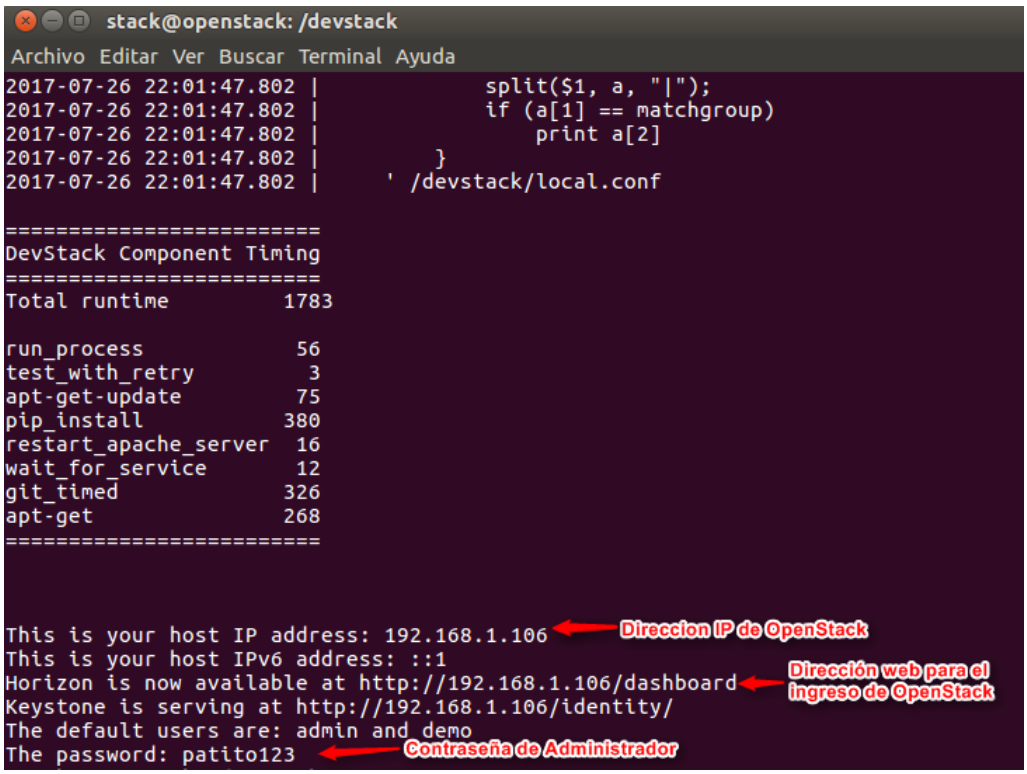

Figura 66 Detalles de la Instalación de OpenStack

8. Ingresar mediante nuestro navegador favorito a la siguiente dirección:

    - http://192.168.1.106/dashboard

9. Digitar las credenciales correspondientes de OpenStack en la pantalla, y a continuación observaremos la pantalla principal, tal como la Figura 67.

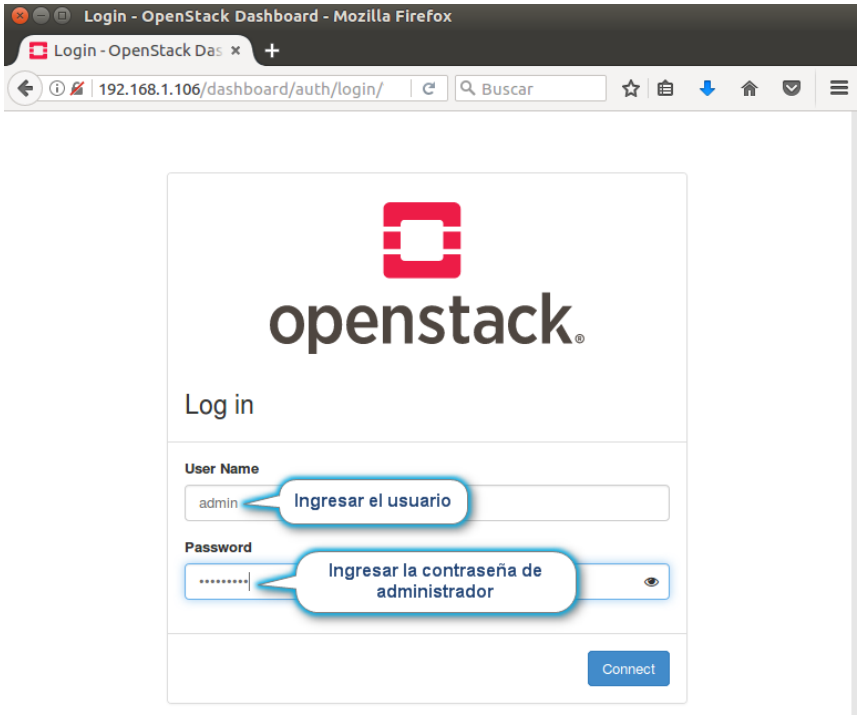

Figura 67 Pantalla de Inicio de Sesión de OpenStack

10. Para la administración y configuración de Openstack, debemos utilizar el usuario **admin,** como se muestra en la Figura 68.

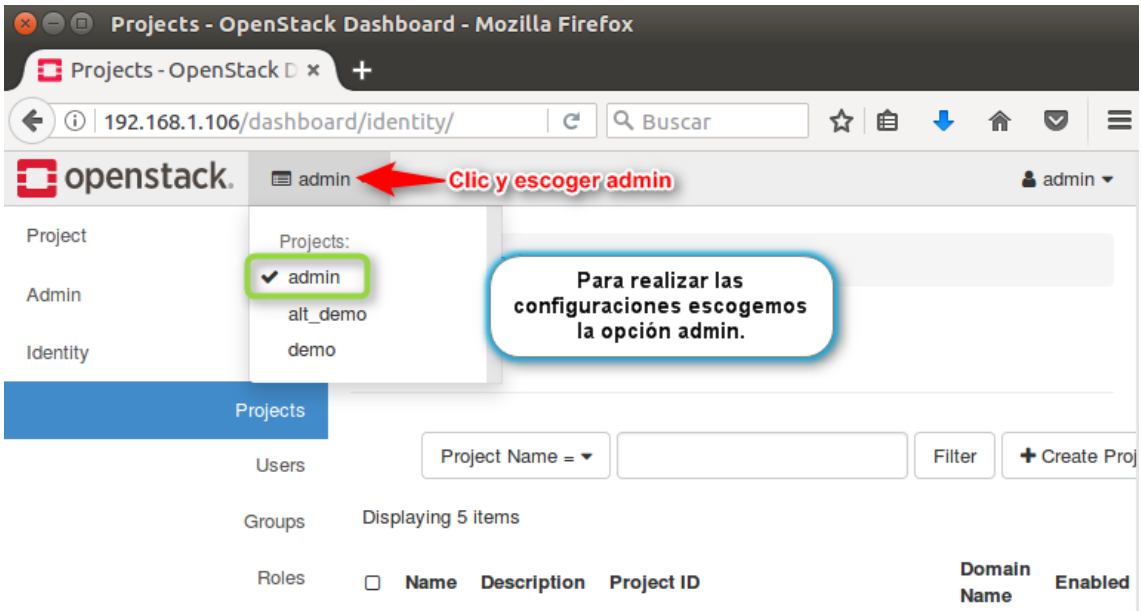

Figura 68 Elegir admin para Administrar OpenStack

Configuración OpenStack

Creación de un Router Neutrón

Este router nos ayudara en el vínculo entre las instancias y la red externa.

Para la creación del router debemos dirigirnos:

a) Project
b) Network
c) Router
d) Create Router

Como se observa en la Figura 69.

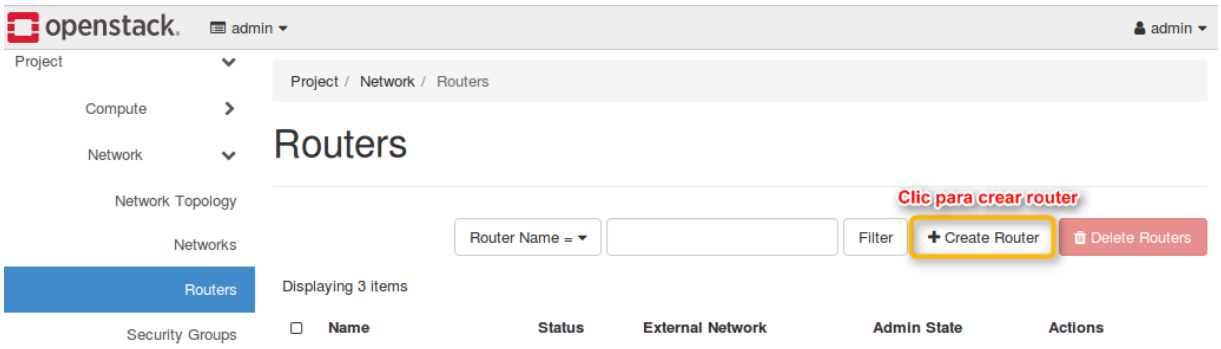

Figura 69 Crear del Router Neutrón en OpenStack

Al seleccionar *Create Router* se nos desplegará una pantalla como la Figura 70, donde se escribira un nombre para el router y escoger la red en este caso *public.*

Figura 70 Parámetros de Configuración del Router en OpenStack

Al finalizar la configuración y dar clic en create router, podremos observar en la figura 71 el router creado.

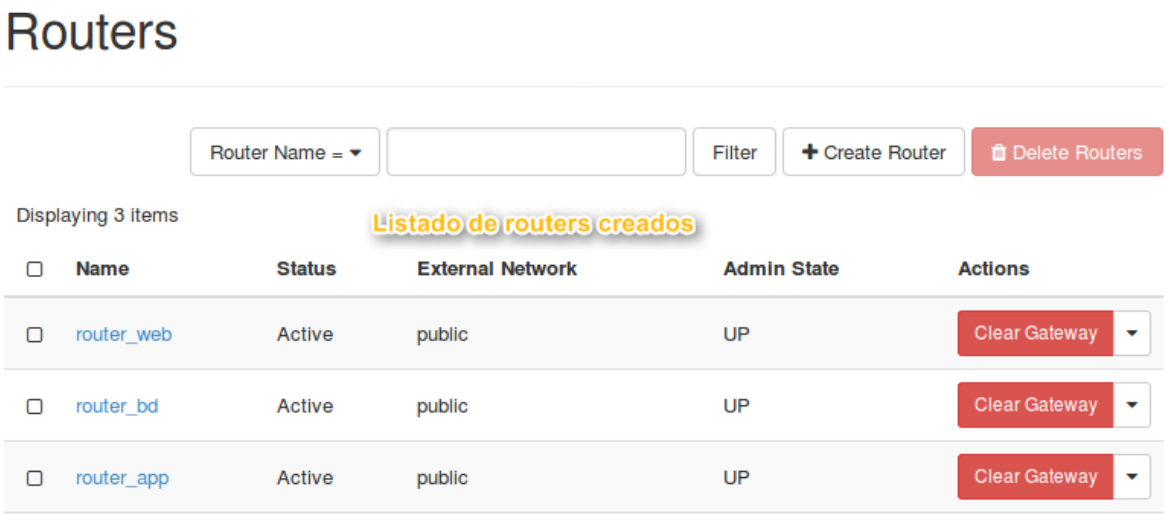

Figura 71 Listado de Routers creados en OpenStack

Creación de una Red Privada

La red privada nos permitirá la asignación IP a la instancia y la conexión entre redes.

Para la creación de la red debemos dirigirnos:

a) Project
b) Network
c) Networks
d) Create Network

Tal como se visualiza en la Figura 72.

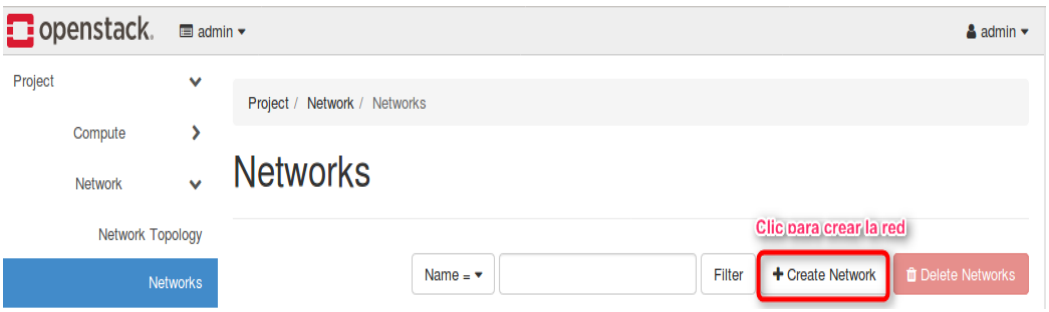

Figura 72 Creación de Redes en OpenStack

Asignar un nombre y seleccionar la opción *Enable Admin State. Como se observa en la Figura 73.*

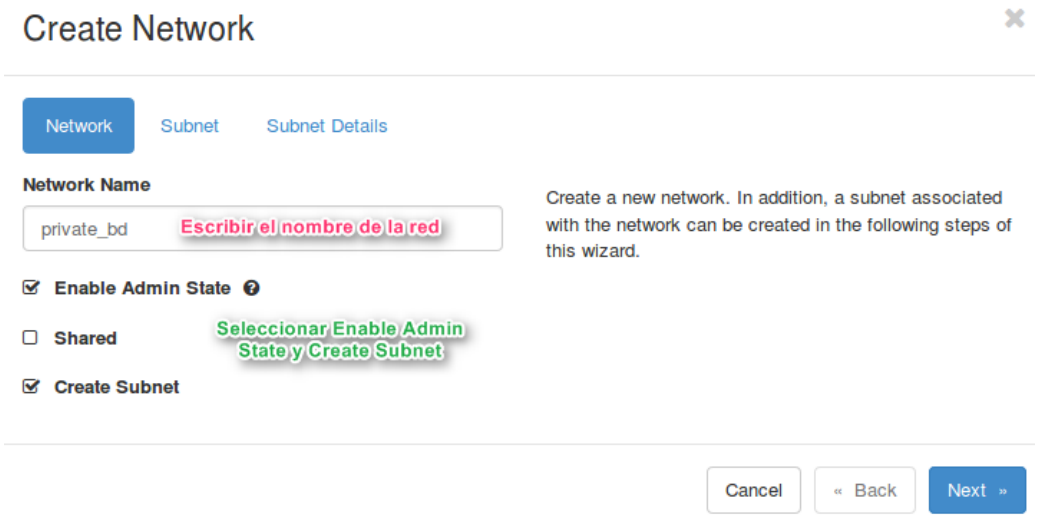

Figura 73 Configuración de parámetros de la red en OpenStack

Creación *de una Subred*

Para la creación de la subred, se procede a escribir un nombre, la dirección de red, el protocolo de internet que vamos a usar (IPv4 o IPv6) y el gateway de la red. Estos parámetros en nuestro caso va estar en IPv6, como se muestra en la Figura 75.

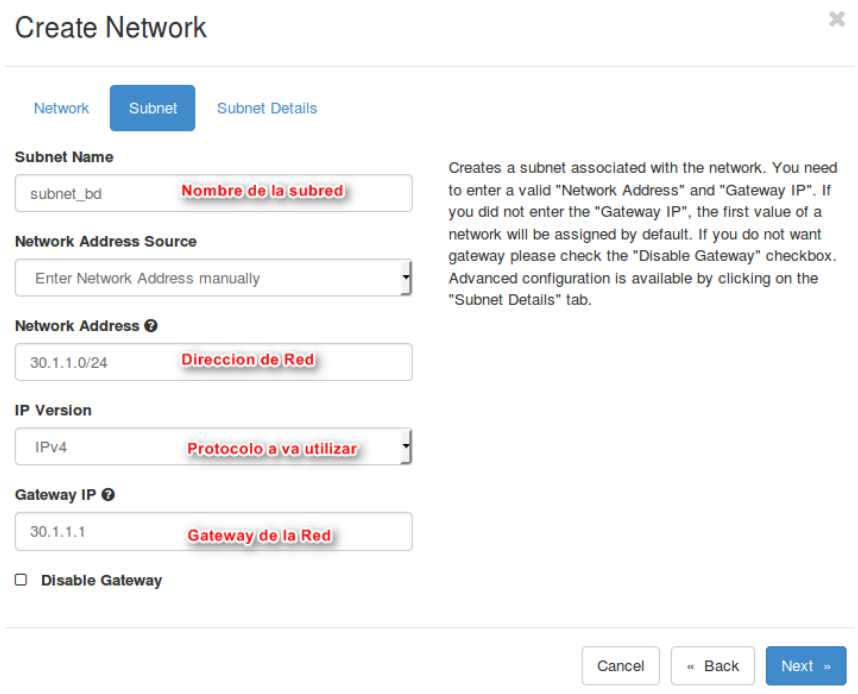

Figura 74 Parámetros para crear la subred en IPv4 en OpenStack

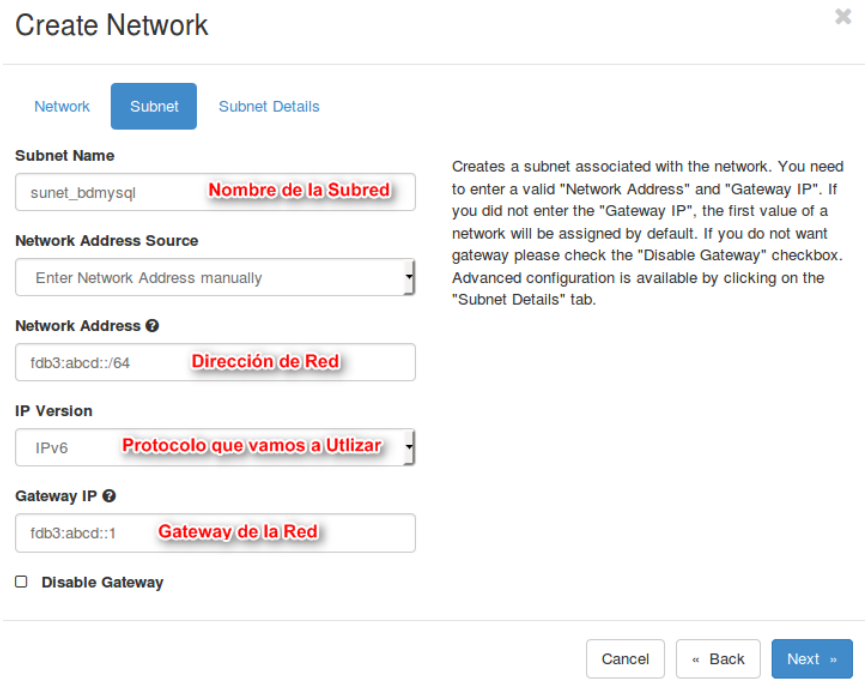

Figura 75 Parámetros para crear la subred en IPv6 en OpenStack

Creación de un pool DHCP

Crear el pool DHCP que sirve para la asignación de direcciones IP, para las instancias que se tiene establecidas en él OpenStack. En este caso se muestra en la Figura 77 en IPv6.

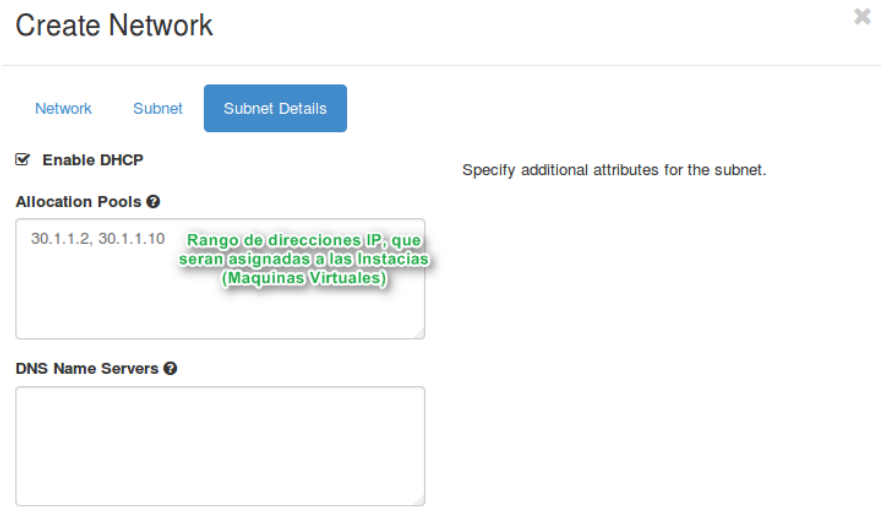

Figura 76 Parámetro del pool DHCP de direcciones IPv4 en OpenStack

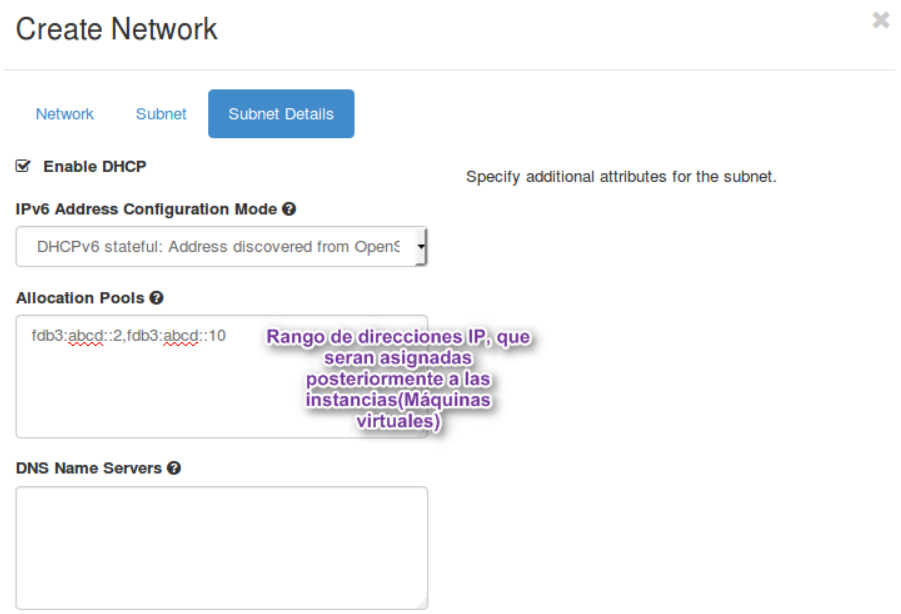

Figura 77 Parámetro del pool DHCP de direcciones IPv6 en OpenStack

Al finalizar la configuración y dar clic en create network, tendremos la red creada con red pública y otra de tipo privada que creamos, debemos verificar que el estado este activo y levantado, como se observa en la Figura 79 en IPv6..

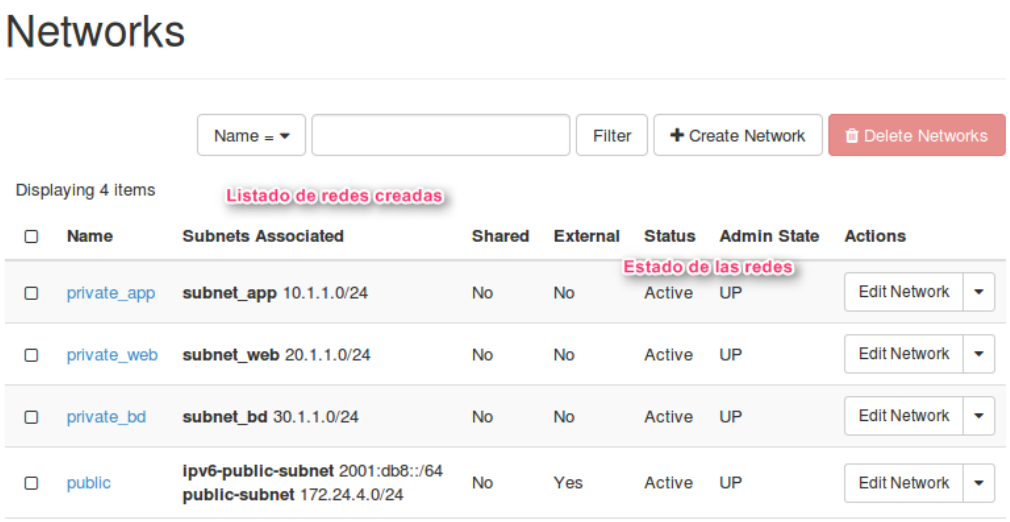

Figura 78 Listado y estados de las redes creadas en IPv4 en OpenStack

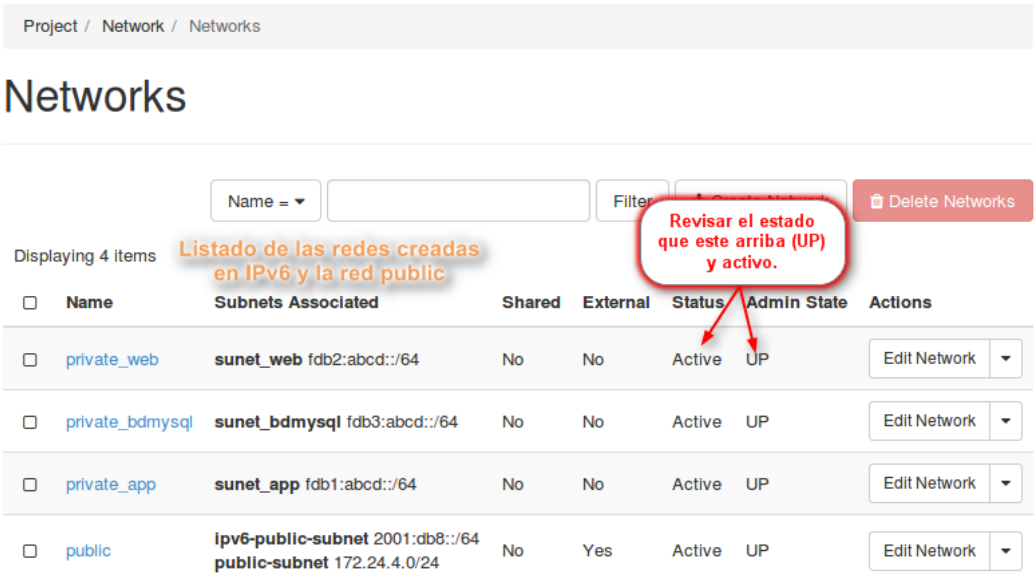

Figura 79 Listado y estados de las redes creadas en IPv6 en OpenStack

Creación de una Imagen

Para la creación se debe subir una imagen vdi o qcow2, aquí debe encontrarse el sistema operativo, en nuestro caso subiremos tres máquinas virtuales de Centos7. Estas máquinas serán cargadas en el OpenStack.

Para la creación debemos ir:

a) Project
b) Compute
c) Images
d) Create Image

Tal como se presenta en la Figura 80.

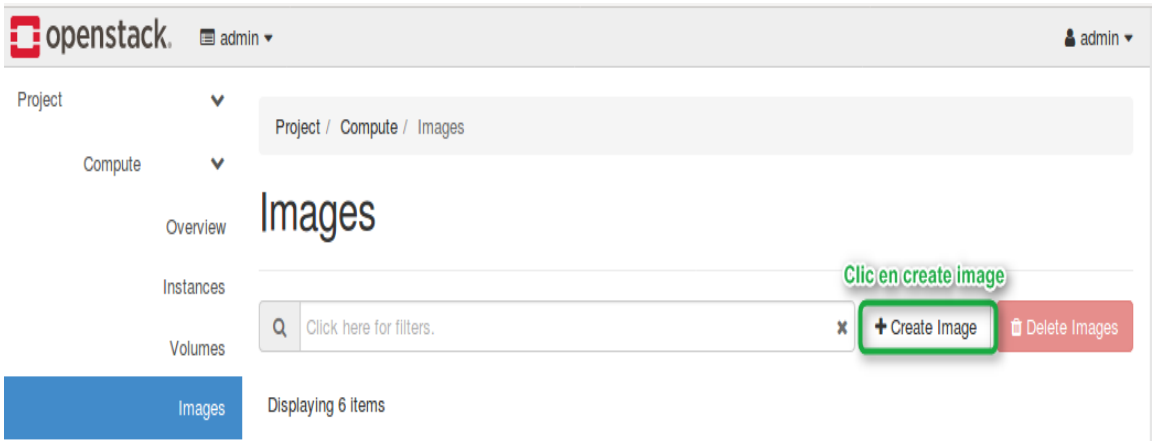

Figura 80 Creación de imágenes en OpenStack en OpenStack

Luego de seleccionar crear imagen, procedemos a llenar los parámetros: nombre de la imagen, seleccionar la imagen (vdi o qcow2), elegir el formato de la imagen, el tamaño del disco y memoria RAM. Como se muestra en la Figura 81.

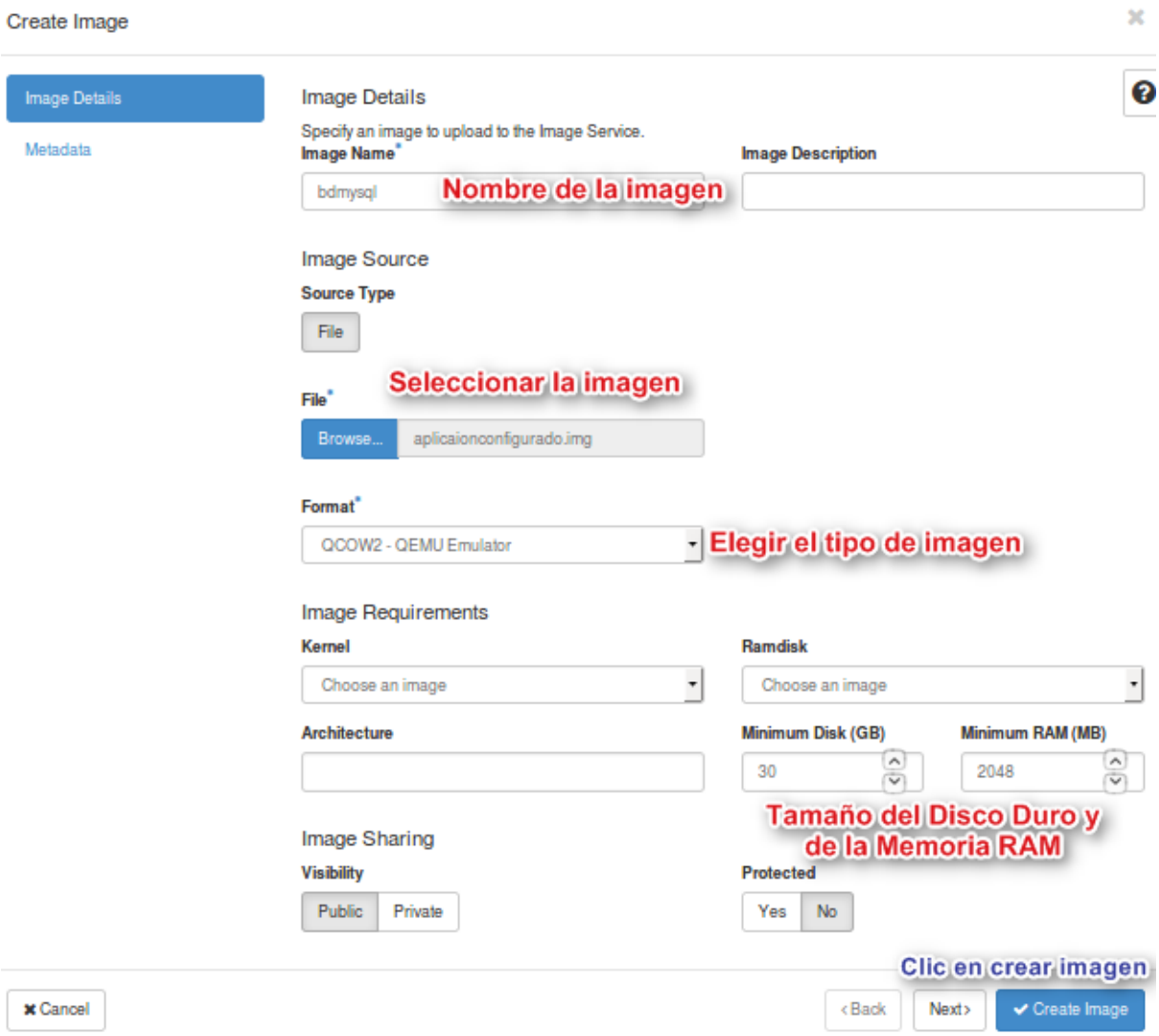

Figura 81 Parámetros para la Creación de la Imagen en OpenStack

Luego del proceso de creación de la imagen, si todo fue correcto y no hubo ningún error, se mostrara un listado de todas las imágenes existentes, tal y como se observa en la Figura 82.

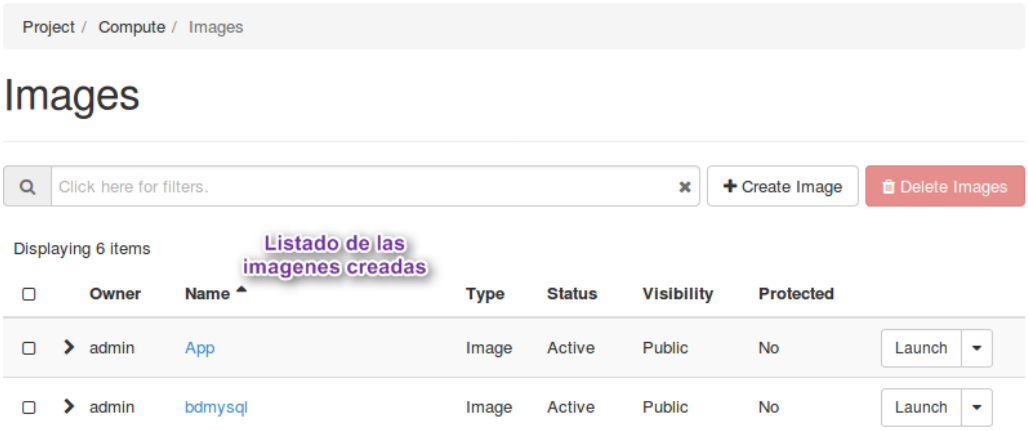

Figura 82 Listado de Imágenes existentes en OpenStack

Creación de llaves para las instancias

Antes crear al instancias (máquinas virtuales), se debe crear una llave para el ingreso mediante ssh para cada una de las.

Para la creación de llaves debemos elegir:

a) Project
b) Compute
c) Key Pairs
d) Create Key Pair

Se nos abrirá una pantalla como la Figura 83, donde debemos proporcionar un nombre para *Create Key Pair*

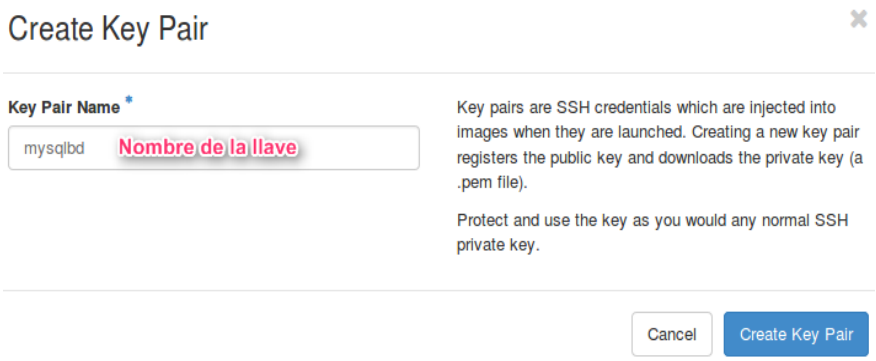

Figura 83 Creación de la llave para las instancias en OpenStack

Nota: Al crear la llave nos pedí que la descarguemos, seleccionamos la ubicación y la guardamos.

Una vez creada y descargada la llave, nos muestra las llaves que han sido creadas, como en la Figura 84 siguiente:

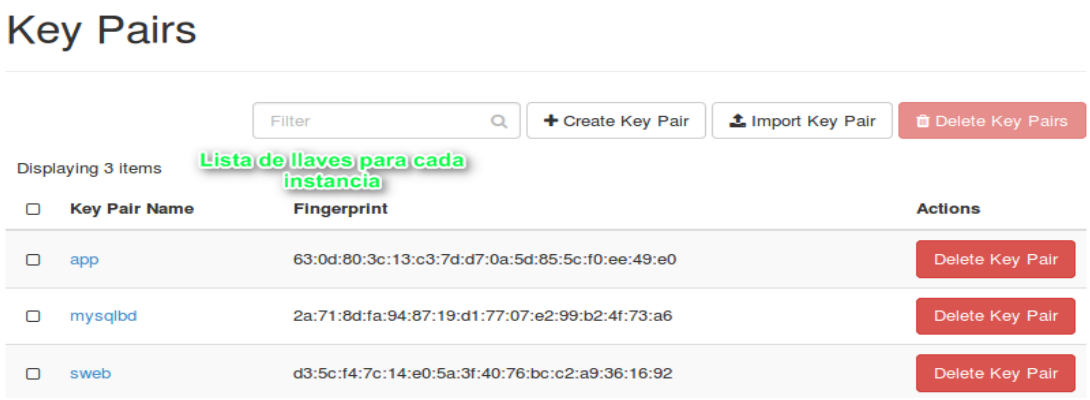

Figura 84 Lista de llaves existentes en OpenStack

Creación de políticas de Seguridad.

La creación de políticas de seguridad, es muy importante y esencial para el funcionamiento correcto de los servicios y acceso a las instancias.

Para configurar las políticas, debemos ir:

a) Project
b) Network
c) Security Groups
d) Manage Rules

Como se visualiza la Figura 85.

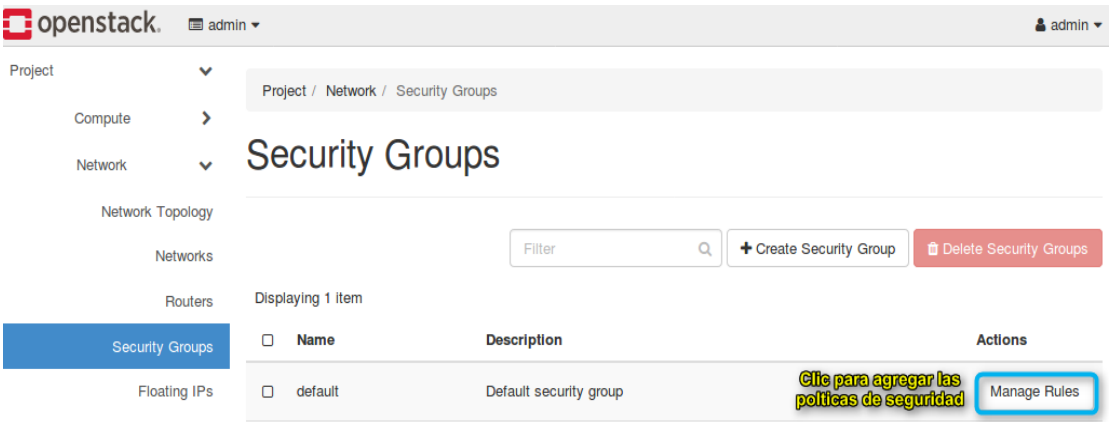

Figura 85 Creación de las Políticas de Seguridad en OpenStack

A continuación seleccionamos *"Add Rule"* y nos aparecerá una pantalla de configuración como la Figura 86. Aquí agregamos los permisos y políticas que necesitemos.

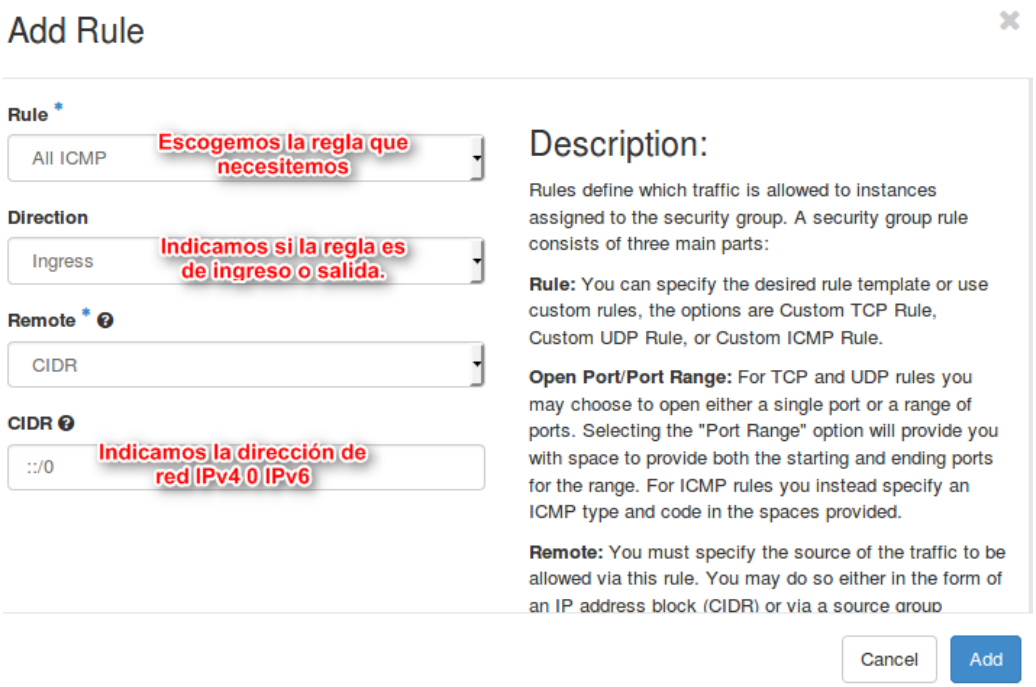

Figura 86 Agregar los Permisos y Politicas de Seguridad en OpenStack

Definimos las reglas que vamos a necesitar para nuestras instancias, como se observa en la Figura 87.

Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Actions
Ingress	IPv6	Any	Any	-	default	Delete Rule
Egress	IPv4	Any	Any	0.0.0.0/0	-	Delete Rule
Ingress	IPv4	Any	Any	-	default	Delete Rule
Egress	IPv6	Any	Any	::/0	-	Delete Rule
Ingress	IPv6	ICMP	Any	::/0	-	Delete Rule
Ingress	IPv6	TCP	22 (SSH)	::/0	-	Delete Rule
Ingress	IPv6	TCP	80 (HTTP)	::/0	-	Delete Rule
Ingress	IPv6	TCP	443 (HTTPS)	::/0	-	Delete Rule
Ingress	IPv6	TCP	3306 (MYSQL)	::/0	-	Delete Rule
Ingress	IPv6	TCP	8009	::/0	-	Delete Rule

Figura 87 Listado de políticas de seguridad en OpenStack

Creación de una Instancia

Luego de crear la llave y las políticas de seguridad, lo siguiente es crear e iniciar las instancias con sus respectivos parámetros.

Para crear la instancia debemos dirigirnos:

1. Project
2. Compute
3. Instances
4. Seleccionamos Launch Instance

Como se muestra en la Figura 88.

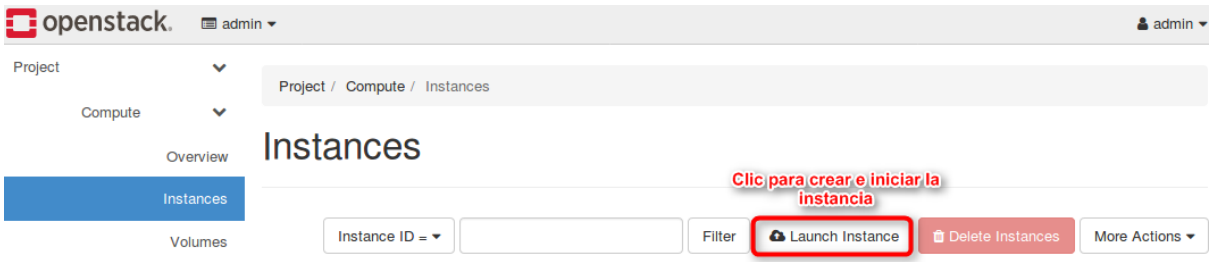

Figura 88 Creación de Instancias en OpenStack

Configuración de Parámetros de una Instancia

Para configurar los diferentes parámetros para crear la instancia, especificamos lo siguiente:

a. Nombre de la Instancia, como se presenta en la Figura 89.

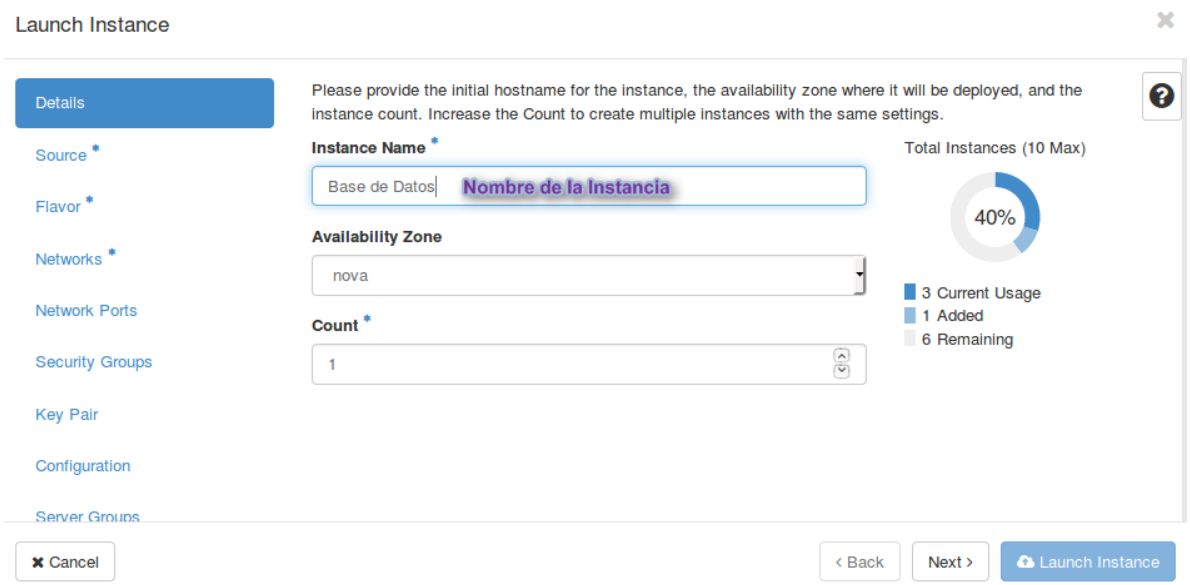

Figura 89 Ingreso del nombre de la Instancia en OpenStack

b. Seleccionar la imagen (vdi o qcow2) de la nueva instancia. Tal como se observa en la Figura 90.

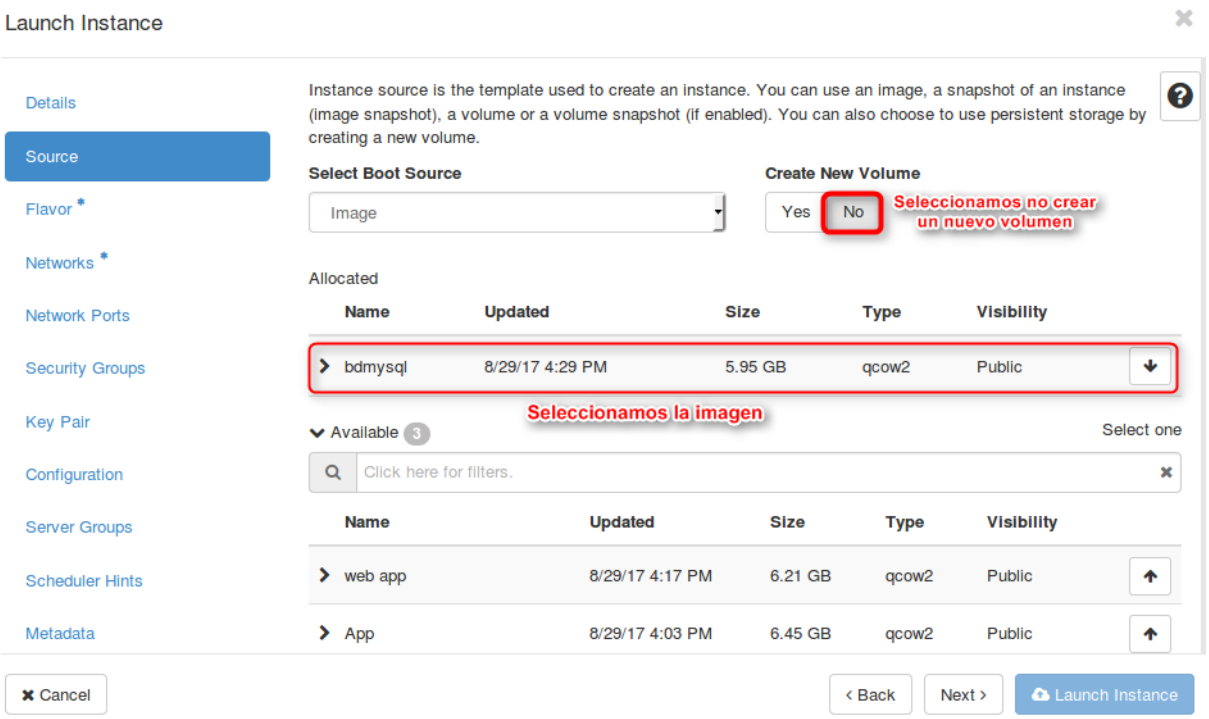

Figura 90 Selección de la Imagen para la nueva instancia en OpenStack

c. Seleccionar Flavor (Sabor), contiene los datos específicos del disco duro y de la memoria RAM. Como se observa en la Figura 91.

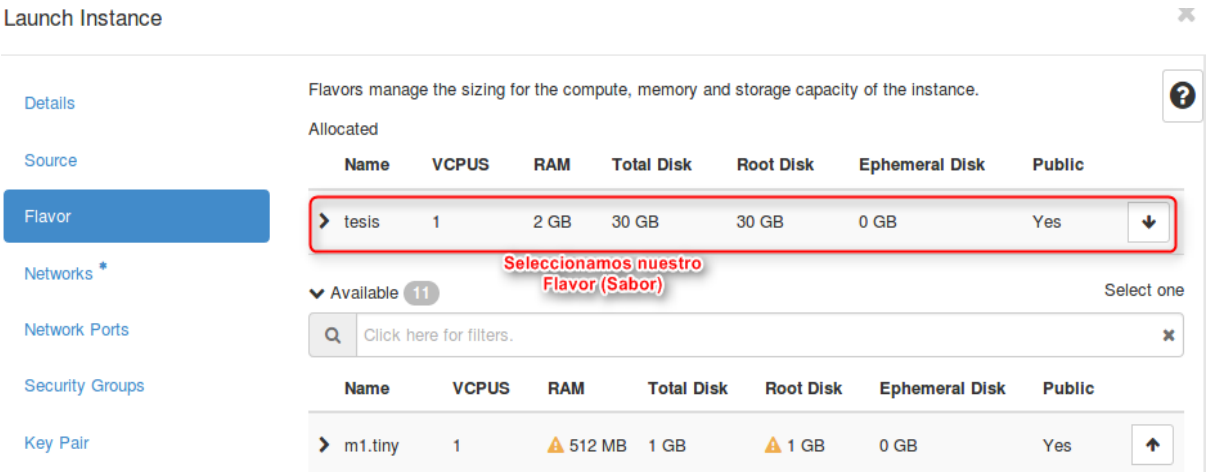

Figura 91 Selección del Flavor (Sabor) en OpenStack

d. Seleccionar la red privada a la que se conectara nuestra instancia. Como se visualiza en la Figura 92.

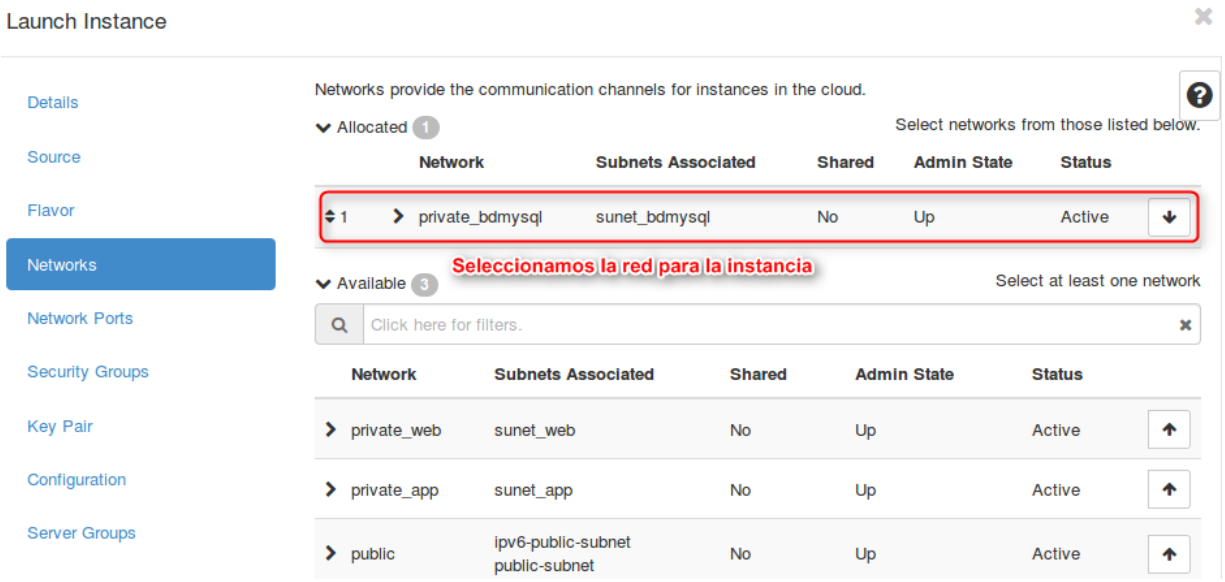

Figura 92 Selección la red privada para la Instancia en OpenStack

e. Las políticas de seguridad dejamos de forma predeterminada, para que wizard de OpenStack nos cargue. Tal como se presenta en la Figura 93.

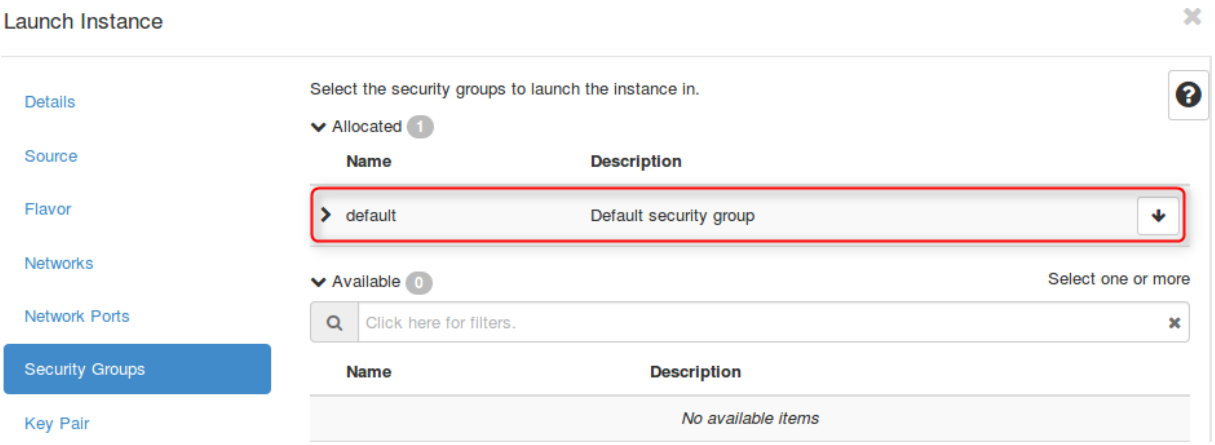

Figura 93 Asignación de Políticas de Seguridad para la Instancia en OpenStack

f. Elegir la llave para la instancia correspondiente. Como se observa en la Figura 94.

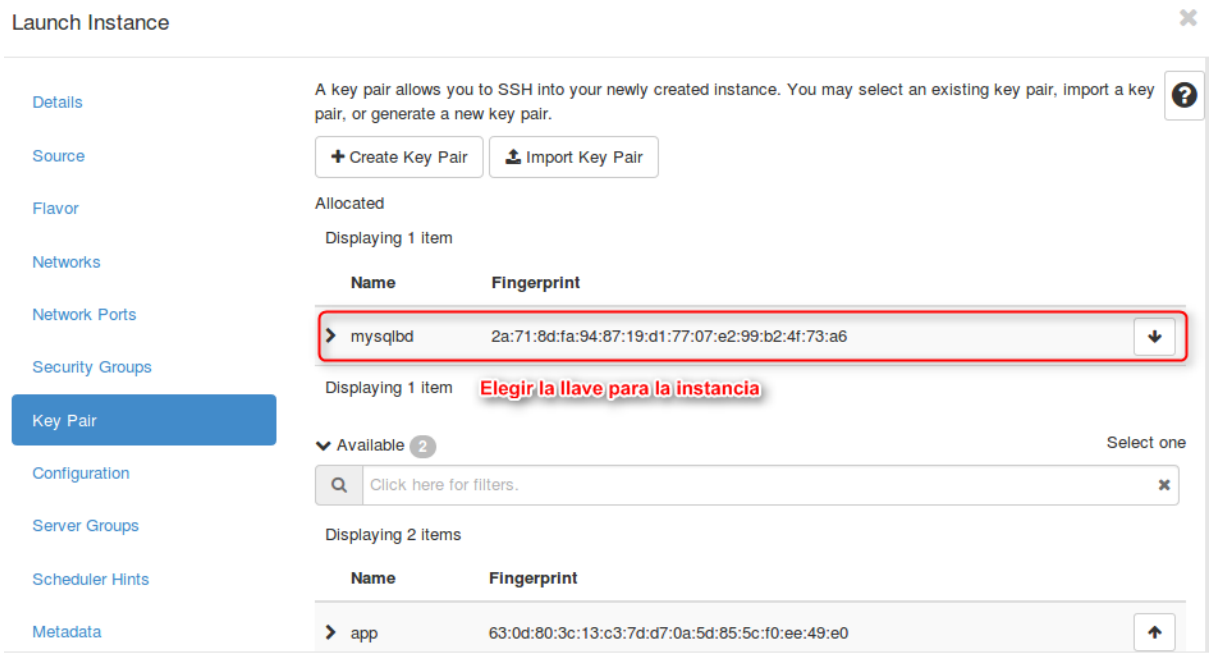

Figura 94 Elección de la llave para la instancia en OpenStack

g. Ya completa la configuración de los parámetros, crear la instancia e iniciar presionando *"Launch Instance",* como se visualiza en la Figura 95.

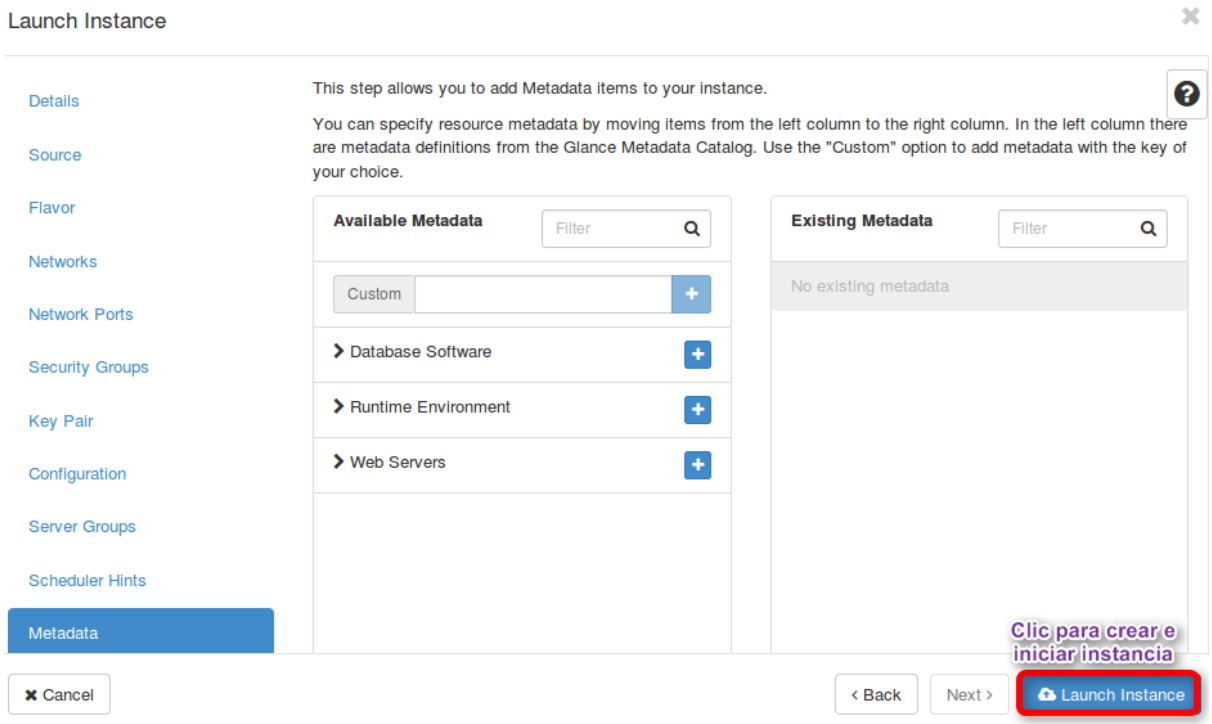

Figura 95 Creación e inicio de la Instancia en OpenStack

h. Una vez creada y ejecutada la instancia, se puede observar en la Figura 97 en IPv6 que está funcionando correctamente, y debemos constatar que el estado del Power State este en ejecución.

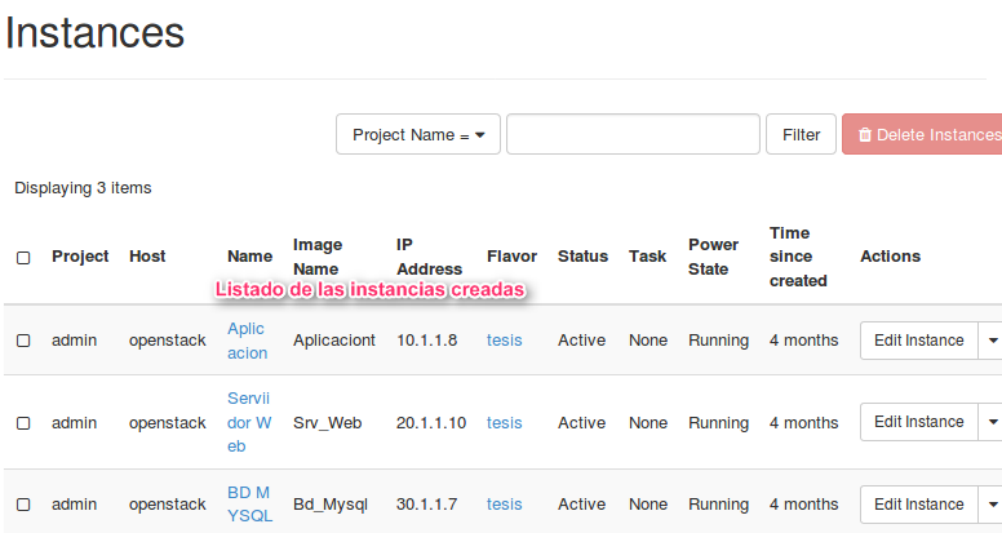

Figura 96 Verificación y listado de las Instancias en IPv4 OpenStack

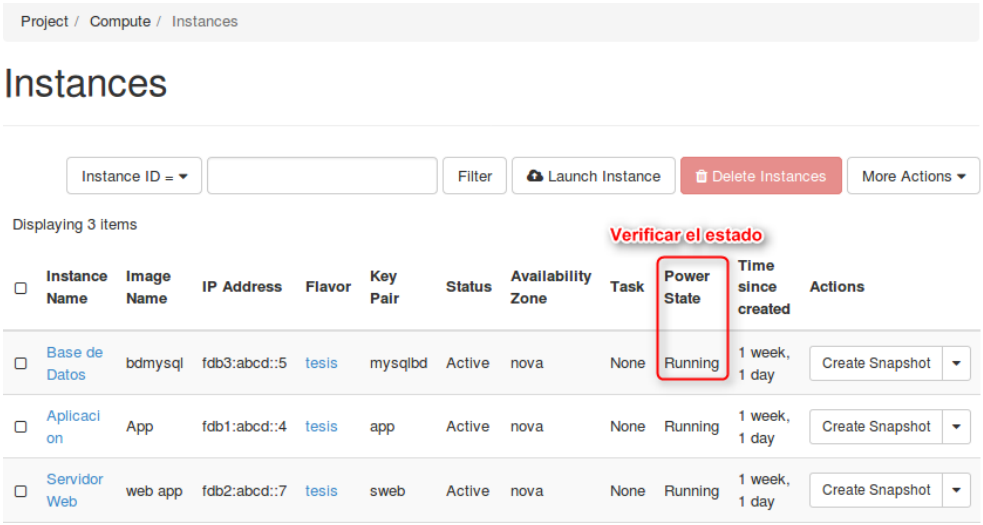

Figura 97 Verificación y listado de las Instancias en IPv6 OpenStack

Verificación de la red creada en OpenStack

Realizada la configuración y ejecución de los routers, las redes locales, políticas de seguridad y nuestras instancias. Podremos visualizar nuestra estructura de red con los Routers Neutrón y las instancias generadas.

Para visualizar las topologías, nos dirigimos:

1. Project
2. Network Network Topology
3. Escogemos “Topology” o “Graph”

En la Figura 98 en IPv4 y 99 en IPv6, podremos observar la topología de los tres routers Neutrón, cada uno con su respectiva instancia, su red privada y la red pública.

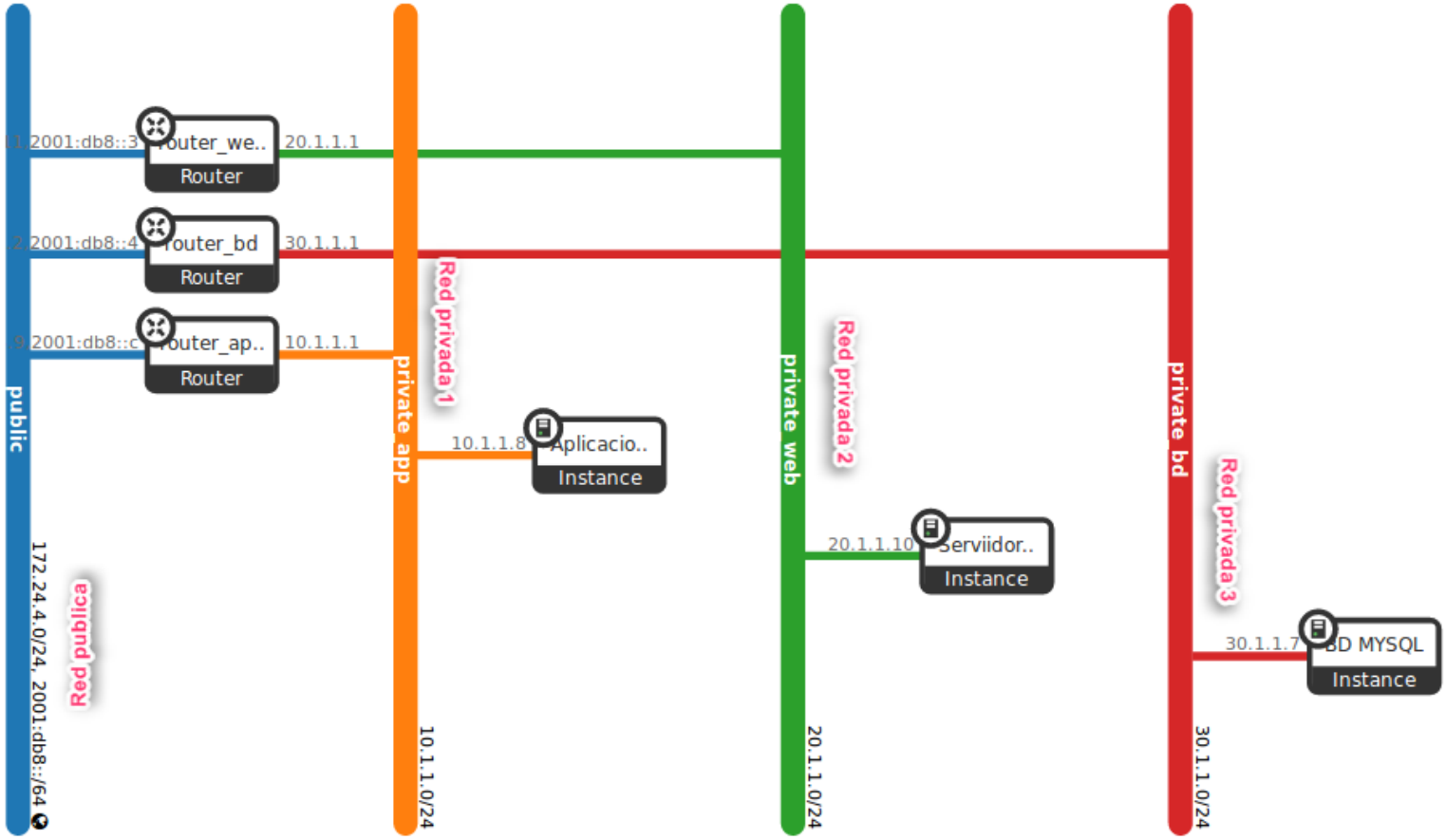

Figura 98 Topología de Red, Routers, Instancias y Redes (Pública y Privada) en IPv4

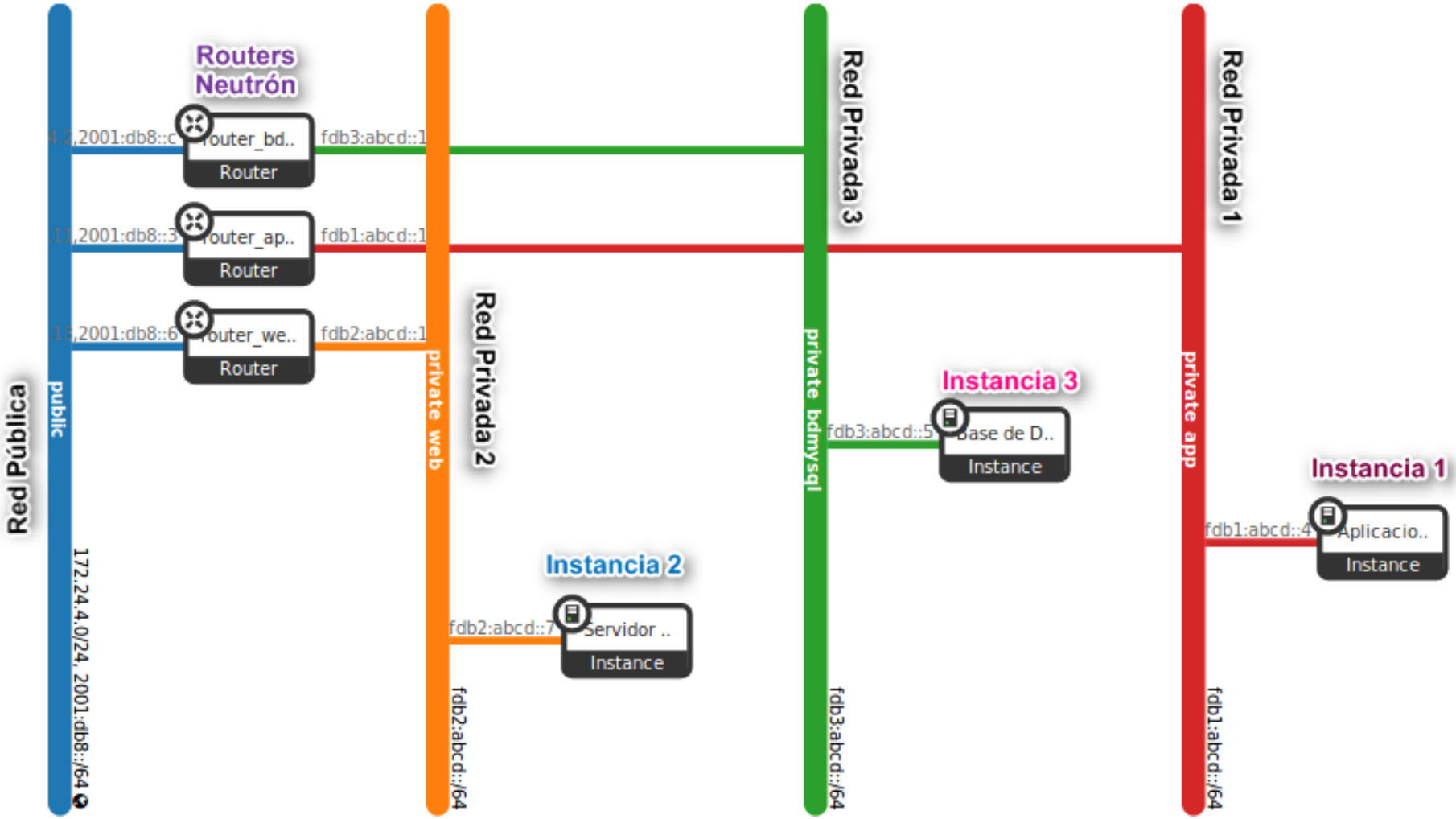

Figura 99 Topología de Red, Routers, Instancias y Redes (Pública y Privada) en IPv6

En la Figura 100 se muestra de manera gráfica la topología de OpenStack, ahí se visualiza el Neutrón (Enrutador), las instancias que están representadas por una computadora, la red privada por una nube y la pública que se conecta directamente al enrutador.

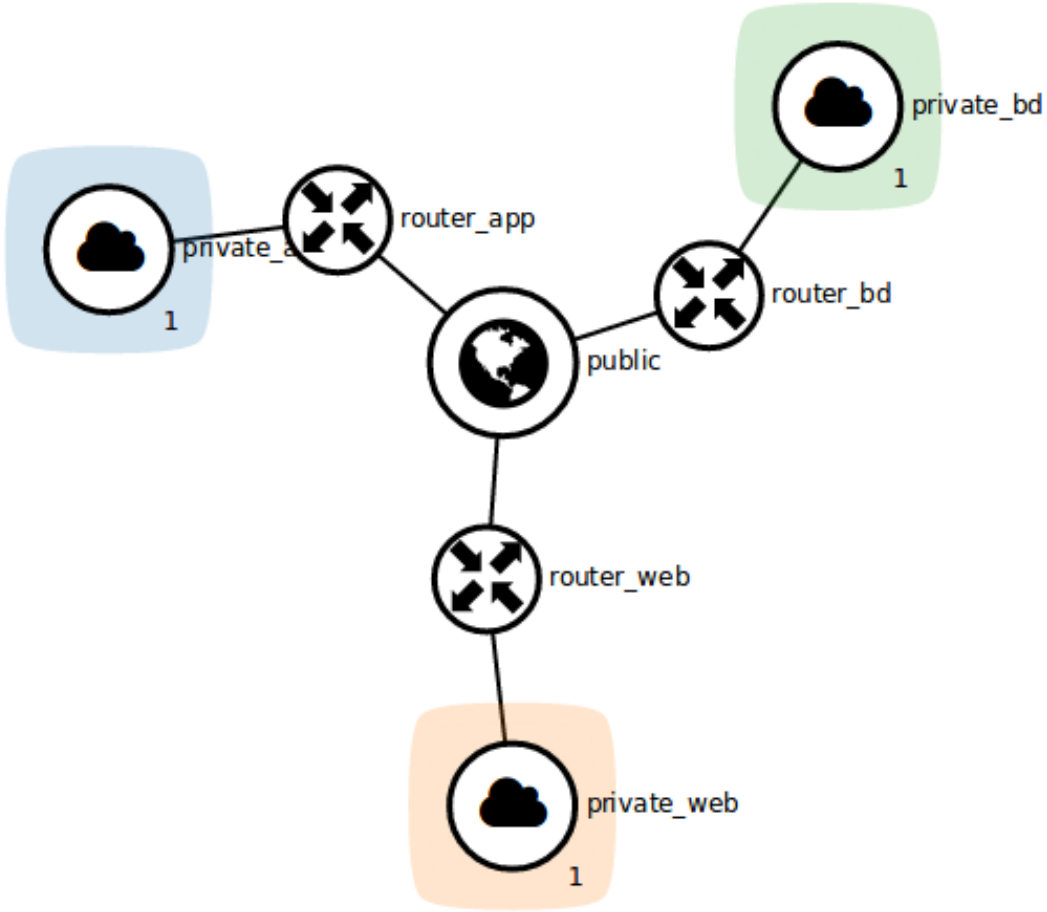

Figura 100 Topología Gráfica de Red en OpenStack

En una terminal escribimos el comando *ifconfig,* se observa en la Figura 101 una interfaz de red llamada *"br-ex",* perteneciente al Router Neutrón a la cual se conectarán los dispositivos de la red externa.

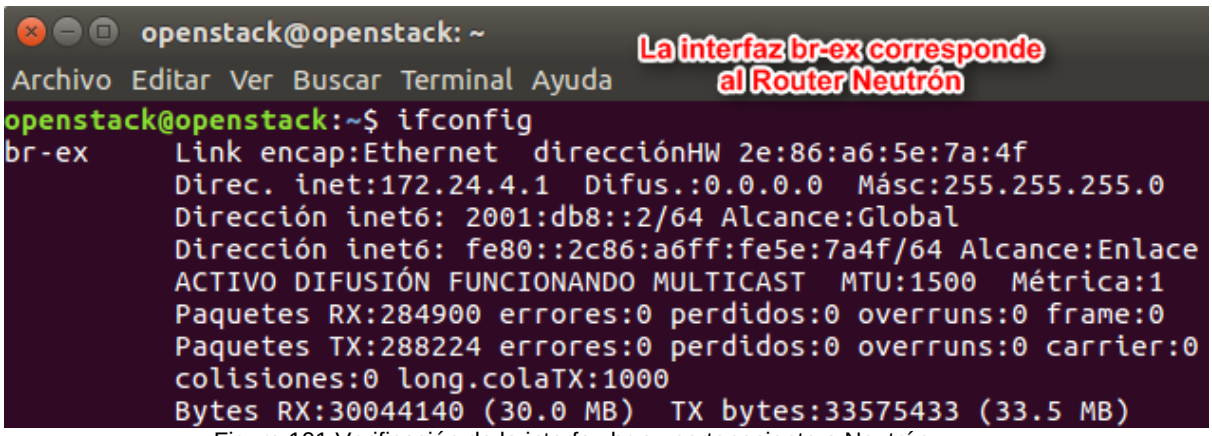

Figura 101 Verificación de la interfaz br-ex perteneciente a Neutrón

Enrutamiento en IPv6

Para establecer la conexión entre la maquina la red pública y privada previamente se debe agregar rutas mediante el Gateway de cada uno de los **Routers.** Como podemos observar en la Figura 102.

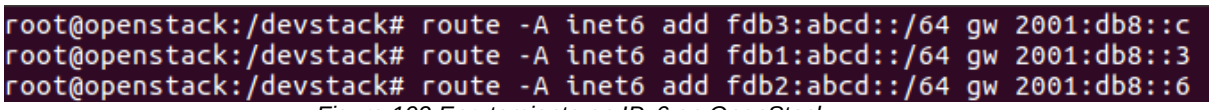

Figura 102 Enrutamiento en IPv6 en OpenStack

Para comprobar si se han agregado correctamente las rutas, mediante el comando "route –A ipv6" podremos visualizar la tabla de enrutamiento en la Figura 103.

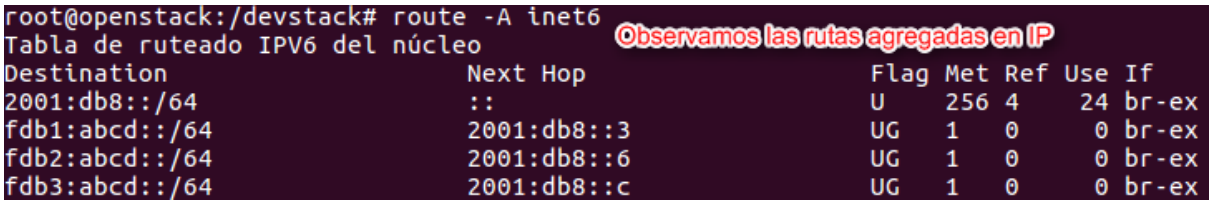

Figura 103 Tabla de enrutamiento en OpenStack