UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y

REDES DE COMUNICACIÓN

TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE

INGENIERO EN ELECTRÓNICA Y REDES DE COMUNICACIÓN

TEMA

IMPLEMENTACIÓN DE UN CLÚSTER-CONTROLADOR DE SDN BASADO EN UN FRAMEWORK DE SOFTWARE LIBRE PARA LA INFRAESTRUCTURA CLOUD DE LA FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS.

AUTOR: CARLOS EDUARDO SANDOVAL CHICAIZA

DIRECTOR: Ing. CARLOS VÁSQUEZ, MSc.

Ibarra- Ecuador

2018

UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN

A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

La Universidad Técnica del Norte dentro del proyecto Repositorio Digital Institucional, determinó la necesidad de disponer de textos completos en formato digital con la finalidad de apoyar los procesos de investigación, docencia y extensión de la Universidad.

Por medio del presente documento dejo sentada mi voluntad de participar en este proyecto, para lo cual pongo a disposición la siguiente información.

DATOS DEL CONTACTO	
Cédula de identidad	040132594-9
Apellidos y Nombres	Sandoval Chicaiza Carlos Eduardo
Dirección	Ibarra – Lic. Nelson Dávila / El Olivo
E-mail	cesandovalc@utn.edu.ec / cesc92.CS@gmail.com
Teléfono fijo	06-2236663
Teléfono móvil	0983373708
DATOS DE LA OBRA	
Título	Implementación de un Clúster-Controlador de SDN basado en un Framework de Software Libre para la infraestructura Cloud de la Facultad de Ingeniería en Ciencias Aplicadas.
Autor	Sandoval Chicaiza Carlos Eduardo
Fecha	14 de Febrero de 2018
Programa	Pregrado
Título	Ingeniera en Electrónica y Redes de Comunicación
Director	Ing. Carlos Vásquez, MSc.

2.AUTORIZACIÓN A FAVOR DE LA UNIVERSIDAD

Yo, Carlos Eduardo Sandoval Chicaiza, con cédula de identidad Nro. 040132594-9, en calidad de autor y titular de los derechos patrimoniales de la obra o trabajo de grado descrito anteriormente, hago entrega del ejemplar respectivo en forma digital y autorizo a la Universidad Técnica del Norte, la publicación de la obra en el Repositorio Digital Institucional y uso del archivo digital en la Biblioteca de la Universidad con fines académicos, para ampliar la disponibilidad de material y como apoyo a la educación, investigación y extensión, en concordancia con la ley de Educación Superior Artículo 144.

3. CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Firma:

Nombre: Sandoval Chicaiza Carlos Eduardo

Cédula: 400132594-9

Ibarra, Febrero de 2018

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE GRADO A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

Yo, Sandoval Chicaiza Carlos Eduardo con cédula de identidad número 040132594-9 manifiesto mi voluntad de ceder a la Universidad Técnica del Norte los derechos patrimoniales consagrados en la Ley de Propiedad Intelectual del Ecuador artículos 4, 5 y 6, en calidad de autor del trabajo de grado con el tema: "IMPLEMENTACIÓN DE UN CLÚSTER-CONTROLADOR DE SDN BASADO EN UN FRAMEWORK DE SOFTWARE LIBRE PARA LA INFRAESTRUCTURA CLOUD DE LA FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS". Que ha sido desarrollado con el propósito de obtener el título de Ingeniero en Electrónica y Redes de Comunicación de la Universidad Técnica del Norte, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En mi condición de autor me reservo los derechos morales de la obra antes citada. En concordancia suscribo en el momento que hago entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Técnica del Norte.

Sandoval Chicaiza Carlos Eduardo

040132594-9

Ibarra, Febrero de 2018

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICACIÓN

Ing. CARLOS VÁSQUEZ, DIRECTOR DEL PRESENTE TRABAJO DE TITULACIÓN

CERTIFICA

Que, el presente Trabajo de Titulación "IMPLEMENTACIÓN DE UN CLÚSTER-CONTROLADOR DE SDN BASADO EN UN FRAMEWORK DE SOFTWARE LIBRE PARA LA INFRAESTRUCTURA CLOUD DE LA FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS" Ha sido desarrollado por el señor Sandoval Chicaiza Carlos Eduardo bajo mi supervisión.

Es todo en cuanto puedo certificar en honor a la verdad.

Ing. Carlos Vásquez. MSc.

DIRECTOR

UNIVERSIDAD TÉCNICA DEL NORTE

UNIVERSIDAD ACREDITADA RESOLUCIÓN 002-CONEA-2010-129-DC
Resolución N° 001-073-CEAACES-2013-13

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

RPC-SO-31-No 573-2016

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y REDES DE COMUNICACIÓN

(No vigente, habilitado para registro de títulos)

Ing. Daniel Jaramillo V.
COORDINADOR

C E R T I F I C O:

QUE, he recibido el producto final que consiste en la implementación de un Cluster controlador de SDN, el cual forma parte del proyecto de titulación: IMPLEMENTACIÓN DE UN CLUSTER – CONTROLADOR DE SDN BASADO EN UN FRAMEWORK DE SOFTWARE LIBRE PARA LA INFRAESTRUCTURA CLOUD DE LA FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS, del señor Carlos Eduardo Sandoval Chicaiza.

Es todo cuanto se puede certificar; faculto al interesado hacer uso del presente en los fines que estime conveniente, excepto en trámites judiciales.

Atentamente,

CIENCIA Y TÉCNICA AL SERVICIO DEL PUEBLO

UNIVERSIDAD TECNICA DEL NORTE
CITEL COORDINACIÓN
CARRERA DE INGENIERIA EN TELECOMUNICACIONES

Ing. Daniel Jaramillo

Ibarra, 07 febrero del 2018

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

AGRADECIMIENTO

A mis padres Aida y Carlos, por todo el esfuerzo, preocupación y atención durante toda mi vida estudiantil, puesto que mis logros son gracias a su ayuda incondicional. Ellos han representado siempre el motor que mueve mi vida y me han motivado a avanzar pese a los duros obstáculos y circunstancias adversas en el camino hacia mi vida profesional.

A mi familia, en especial a Daniel que con sus palabras de aliento y apoyo moral han sabido brindarme la fuerza para no rendirme nunca, haciendo que no desfallezca y siga adelante. A mis abuelos que siempre me han inspirado con su ejemplo de honradez, responsabilidad y perseverancia.

Un sincero y especial agradecimiento a mi director de tesis, el Ing. Carlos Vásquez por la orientación en el transcurso de mi paso por la vida universitaria y por haberme dado la oportunidad de probar que, no existe tesis imposible y que, con trabajo duro, esfuerzo, constancia y dedicación todo es posible.

Un viejo proverbio chino dice “Cuando bebas agua recuerda la fuente”. A la planta docente de la Facultad de Ingeniería en Ciencias Aplicadas que durante mi formación profesional compartieron conmigo su amistad, sus consejos y algunos de sus más valiosos secretos de ingeniería. A todos, extiendo el más limpio sentimiento de gratitud.

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

DEDICATORIA

Este proyecto de titulación se lo dedico a Carlos mi padre, que, aunque ya no se encuentra a mi lado, su sueño más grande fue verme convertido en un profesional. No existen palabras para expresar mi eterno agradecimiento y espero que donde se encuentre se sienta orgulloso.

RESUMEN

El presente proyecto de tesis muestra el despliegue de una Red Definida por Software construida a base de elementos de software libre. El despliegue plantea una solución a la problemática de la gestión desarticulada de los recursos de networking del Centro de Datos FICA. Como parte de la etapa inicial se ejecuta una reorganización de los componentes y la creación de los elementos de red a base de virtualización para evitar el crecimiento en la cantidad de hardware de red y hacer uso de recursos genéricos creados mediante software.

La aplicación de la arquitectura SDN requiere el manejo de conceptos y definiciones relacionadas con su arquitectura y componentes, así como los términos relacionados con la virtualización. Por esa razón este documento dedica una sección completa al análisis teórico de las alternativas para la implementación del controlador de red y de los modelos de despliegue de las SDN. Se consideran también los métodos de clusterizacion para integrar la infraestructura cloud del Centro de Datos con la red definida por software usando el mecanismo NetVirt.

Uno de los objetivos principales de la etapa de despliegue, es la explicación de los pasos que se siguen para el levantamiento de la SDN. Se facilitan los movimientos detallados de la evaluación del cloud privado y posteriormente se aplican pruebas prácticas a los controladores candidatos, para obtener el resultado con pruebas fehacientes. Para este proyecto se ha hecho uso de las normas ISO/IEC/IEEE 29148 y 2504n para validar el software sobre el que se despliega la SDN.

Luego de las pruebas de funcionamiento en varias sub etapas, se verifica que el funcionamiento de los elementos creados durante despliegue es satisfactorio. Aquí se pone en funcionamiento a toda la red permitiendo comprobar que el despliegue se ha concluido con éxito y que los mecanismos de virtualización de red funcionan están listos para usarse en producción.

ABSTRACT

The present thesis project shows the deployment of a Software Defined Network built on the basis of free software elements. The deployment poses a solution to the problem of the disjointed management of the networking resources of the FICA Data Center. As part of the initial stage, a reorganization of the components and the creation of network elements based on virtualization is executed to avoid the growth in the amount of network hardware and make use of generic resources created by software.

The application of the SDN architecture requires the management of concepts and definitions related to its architecture and components, as well as the terms related to virtualization. For this reason, this document devotes a complete section to the theoretical analysis of the alternatives for the implementation of the network controller and the SDN deployment models. Clustering methods are also considered to integrate the data center cloud infrastructure with the software defined network using the NetVirt mechanism.

One of the main objectives of the deployment stage is the explanation of the steps that are followed for the lifting of the SDN. The detailed movements of the evaluation of the private cloud are facilitated and then practical tests are applied to the candidate controllers, to obtain the result with reliable evidence. For this project, the ISO / IEC / IEEE 29148 and 2504n standards have been used to validate the software on which the SDN is deployed.

After the tests of operation in several sub stages, it is verified that the operation of the elements created during deployment is satisfactory. Here the whole network is put into operation, making it possible to verify that the deployment has been completed successfully and that the network virtualization mechanisms are working and ready to be used in production.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO I

1. ANTECEDENTES

En esta primera sección se facilita una referencia para el análisis sobre el tema desarrollado en la presente memoria del proyecto, en la que se incluyen detalles del estudio previo realizado para el cumplimiento de las etapas que comprenden la ejecución y puesta en marcha según los objetivos que a continuación se plantean.

1.1. TEMA

Implementación de un clúster-controlador de SDN basado en un framework de software libre para la infraestructura Cloud de la Facultad de Ingeniería en Ciencias Aplicadas.

1.2. DEFINICIÓN DEL PROBLEMA

Las tecnologías han evolucionado tan rápido que han convertido a las redes en el punto de falla, impidiéndoles crecer y por consecuencia ya no permiten satisfacer las necesidades tecnológicas de las empresas, operadores y usuarios lo que nos lleva a perseguir un cambio de arquitectura que nos brinde flexibilidad, escalabilidad, automatización y control. (Morreale & Anderson, 2015)

Actualmente los requerimientos de las organizaciones incluyen acortar el espacio físico de los servidores por lo que se recomienda trabajar los servicios de red también sobre la nube. Esto demanda de ciertos requisitos en subcapa que pueden cubrirse mediante el despliegue de una SDN sobre nodos clusterizados; es así que estas exigencias desafían en gran medida a las capacidades que pueden proporcionar y el tráfico que son capaces de soportar los dispositivos de red comunes.

La Facultad de Ingeniería en Ciencias Aplicadas de la Universidad Técnica del Norte en su Centro de Datos ha implementado un cloud privado, el cual está limitado en cuanto a la carga que es capaz de soportar la infraestructura individualmente lo que no representa un escenario optimo,

razón por la que se busca acortar espacios para de esta manera beneficiar a los proyectos de investigación sobre nuevas arquitecturas de red en la facultad que permitan de esta manera atravesar la barrera que establecen las redes tradicionales. Debido a esto se han implementado servidores de aplicaciones mediante soluciones de cloud computing del tipo IaaS, pero de manera dispersa lo que no favorece a la gestión y administración de capacidades y servicios de red.

Optimizar el espacio físico de Data Center y reducir la huella de carbono es uno de los objetivos más importantes en el proceso de migración a nuevas tecnologías ya que esto supone disminuir en gran medida el consumo energético y enfatizar en la ventaja estratégica del diseño y construcción verdes.

1.3. OBJETIVOS

1.3.1. Objetivo General

Desplegar una Red Definida por Software mediante el uso de un clúster de servidores y un framework de código abierto sobre la infraestructura cloud de la Facultad de Ingeniería en Ciencias Aplicadas.

1.3.2. Objetivos Específicos

- Estudiar el estado del arte de SDN y los conceptos de clusterización.
- Establecer la situación actual de la red y análisis de las funcionalidades del controlador.
- Desplegar la red SDN sobre la infraestructura cloud de la facultad.
- Clusterizar los servidores del Centro de Datos que mejore la distribución de las capacidades de la infraestructura cloud de la facultad.
- Efectuar las pruebas de funcionamiento de la red clusterizada garantizando óptima gestión.

1.4. ALCANCE

El presente desarrollo propone la implementación de una Red Definida por Software sobre la infraestructura del Centro de Datos de la Facultad de Ingeniería en Ciencias Aplicadas y la clusterización de los servidores para proporcionar una óptima gestión de los servicios que ofrece el cloud de la misma.

Para iniciar con el proyecto es necesario realizar la fundamentación teórica y estudiar el estado del arte de todos los aspectos relacionados con SDN, así como de los conceptos de clusterización. Dentro de esta etapa también se enmarcará el estudio de temáticas como métodos de despliegue y selección de componentes de hardware y software.

A continuación, se establecerá estado actual del equipamiento y del software de aplicaciones instalado en la red para conocer las limitaciones de la infraestructura y de esta manera efectuar un despliegue óptimo de la red. Por otra parte, se analizará la funcionalidad del controlador para visualizar el alcance del control de la red mediante Open vSwitch.

Una vez que el ambiente se ha preparado se desplegará la Red Definida por Software centralizando la misma en el framework de código abierto que constituirá el controlador de red y permitirá añadir uno por uno los nodos que luego formaran parte del grupo de procesamiento paralelo. De ser necesaria la reorganización de los nodos se hará basándose en sus características y capacidades de manera que se consiga un punto de funcionamiento óptimo.

Posteriormente con la SDN desplegada se realizará pruebas de verificación usando terminales de red u ordenadores para comprobar que la gestión de capacidades en los nodos funciona correctamente y así proceder a la clusterización añadiendo los nodos adicionales al controlador

hasta lograr que todos funcionen simultáneamente y sus características puedan ser gestionadas desde el nodo maestro permitiendo proporcionar una plataforma estable.

Finalmente, con la infraestructura física y lógica desplegada y realizada la puesta en marcha del clúster-controlador se procederá a ejecutar las pruebas de funcionamiento necesarias para determinar que el proyecto se haya culminado con satisfacción y se encuentra en total funcionamiento.

1.5. JUSTIFICACIÓN

Las redes tradicionales han ido evolucionando a lo largo de la historia con el objetivo de satisfacer las diferentes necesidades tecnológicas de las empresas, de los operadores de telecomunicaciones y de los usuarios finales razón por la que se han vuelto muy complejas y de esta manera se ha restado su escalabilidad en gran medida.

Las redes definidas por software (SDN) son una manera de abordar la creación de redes en la cual el control se desprende del hardware y se le da el mismo a una aplicación de software llamada controlador. El termino SDN se ha venido acuñando en los últimos años para hacer referencia a una arquitectura de red que permite separar el plano de control, del plano de datos, para conseguir redes más programables, automatizables y flexibles.

Mediante la centralización de la gestión de capacidades es posible reducir la dependencia del hardware de los sistemas servidores mediante virtualización y a continuación aplicando técnicas de clúster de alta disponibilidad se puede unir tecnologías en un sistema que permita disponer de servicios que estén activos el mayor tiempo posible y con un buen aprovechamiento de los recursos presentes a un menor costo. (VMware, 2017)

Con la adopción del cloud se abren un sinfín de posibilidades de crecimiento permitiendo proveer servicios mediante el concepto de Software como Servicio, así se democratiza el acceso a capacidades de software de primer nivel, dado que una aplicación de software proporciona servicio a varios clientes. El Cloud Computing también puede ser extremadamente seguro y, a menudo, superar los niveles de seguridad de la informática tradicional.

Las nubes y las redes SDN han sido diseñadas para reemplazar en un futuro a las redes tradicionales pero el concepto más indispensable para lograr alta disponibilidad y rendimiento, la clusterización de servidores se mantiene en el mismo nivel de usabilidad. En un clúster, dos o más unidades operan para proporcionar alta disponibilidad, confiabilidad y escalabilidad que no puede obtenerse usando una sola.

El presente proyecto pretende solucionar los problemas de escalabilidad y dispersión de servicios introduciendo el concepto de paralelismo mediante el despliegue de una Red Definida por Software y la clusterización de los servidores disponibles dentro del Centro de Datos de la Facultad de Ingeniería en Ciencias Aplicadas permitiendo optimizar el funcionamiento global de la red proporcionando tolerancia a fallos, disponibilidad continua y escalabilidad masiva.

CAPÍTULO II

2. FUNDAMENTO TEÓRICO

En este capítulo se fundamenta teóricamente algunos conceptos que sirven como guía para la realización del presente proyecto, entre ellos la evolución de las tecnologías de red, limitaciones que presentan las redes actuales, características fundamentales de las redes definidas por software y todos los componentes de la virtualización de redes y servicios en un ambiente clusterizado.

2.1. REDES DEFINIDAS POR SOFTWARE (SDN)

La Open Networking Foundation (ONF) es la organización más relacionada con el trabajo sobre SDN. Según la ONF, Software Defined Networking es una arquitectura más flexible ideal para las aplicaciones actuales que demandan ancho de banda de manera dinámica. Esta arquitectura emancipa las funciones de control y reenvío de datos en la red permitiendo que la red sea directamente programada siendo disociada del hardware subyacente para aplicaciones y servicios. OpenFlow se convierte en el componente principal para la fabricación de soluciones basadas en Redes Definidas por Software de manera ágil, centralizada y basada en estándares abiertos y neutrales. (Open Networking Fundation, 2017)

2.1.1. EVOLUCIÓN DE LAS TECNOLOGÍAS DE RED

Con el paso de los años las redes han presentado cambios que han permitido tener mayor convergencia y mejoras en la interconexión. Se puede considerar que durante ese tiempo han existido distintas tecnologías que han sido elementos clave para la construcción de lo que hoy es la red de redes. Las fases en las que se han incorporado mayores cambios son cinco.

- **Terminales Remotos (Mainframe):** En esta era la conectividad era de naturaleza remota y era necesario trabajar desde un terminal usando periféricos que se comunicaban con el mainframe

central. Las conexiones de red eran limitadas y casi siempre se usaba el esquema punto a punto o punto multipunto.

- **Conexiones Punto a Punto (Peer-to-Peer):** La necesidad de compartir información demandó el desarrollo de nuevos protocolos y nuevos ambientes. Aunque la comunicación siguió siendo punto a punto se mejoraba el aspecto de conectividad permitiendo que la comunicación también sea entre pares de máquinas participantes.

- **Redes de Área Local (LAN):** La evolución informática hacia sistemas independientes implantó la necesidad de desarrollar la tecnología de red de área local que trajo consigo nuevos protocolos, métodos de acceso y la posibilidad de segmentar los dominios de colisión y de difusión.

- **Red Puenteada (Bridged Networks):** La escalabilidad a nivel físico dio lugar a la creación de los dispositivos llamados puentes que con el paso de los años se convertirían en los conmutadores de hoy en día. Gracias a estos dispositivos se hizo posible interconectar múltiples dominios compartidos, permitiendo reenviar paquetes entre múltiples hosts.

- **Redes Enrutadas (Routed Networks):** De la misma amanera que las redes puenteadas la necesidad de redirigir paquetes hacia distintos destinos a grandes distancias dio lugar a la fabricación y uso de enrutadores de paquetes que trabajan a nivel de capa 3 en el modelo OSI. Así mediante protocolos distribuidos se toman las decisiones adecuadas para el reenvío.

2.1.2. LIMITACIONES DE LAS REDES ACTUALES

Las redes de hoy son el resultado de las decisiones sobre protocolos tomadas hace cuatro décadas, para ese entonces se preveía que el uso de 32 bits sería suficiente para cubrir el

direccionamiento de internet que se aproximaba a los 16 millones. No fue hasta febrero de 2011 que IP versión 4 quedó obsoleto y sus direcciones se agotaron. (Morreale & Anderson, 2015)

En un inicio no se esperaba que las redes cambiaran tanto, los servidores que se conectan a la red han sufrido dramáticas transformaciones en los últimos 10 años y con la llegada de la virtualización el papel de un servidor ha cambiado drásticamente. Los servidores ahora son dinámicos, se crean y se mueven fácilmente aumentando al número que se pueden usar en una red.

Antes de la llegada de la virtualización a gran escala, las aplicaciones se asociaban a un servidor único que se encontraba en una ubicación fija de la red. Hoy en día con el uso de máquinas virtuales se puede intercambiar flujos de tráfico optimizando y equilibrando las cargas de trabajo de los servidores creando desafíos para las redes tradicionales relacionados con el direccionamiento, espacios de nombres y segmentación basada en enrutamiento. (Open Networking Fundation, 2017)

2.1.3. CARACTERÍSTICAS FUNDAMENTALES DE SDN

A partir de una serie de propuestas, estándares e implementaciones para mejorar las redes, SDN evoluciona sobre las alternativas como ForCES, 4D y Ethane debido a cinco rasgos fundamentales: separación de planos, simplificación de dispositivos, control centralizado, automatización de red y virtualización.

2.1.3.1. Separación de planos

Una de las características principales que revela SDN es la posibilidad de separar los planos de control y de datos que originalmente en una red de tradicional mantienen una estrecha relación como se muestra en la Figura 1, esto sucede a causa de que los encargados de proporcionar ambos planos son los fabricantes; de esta manera si un conmutador debe calcular una tabla de reenvío es él mismo quien tiene que implementarla.

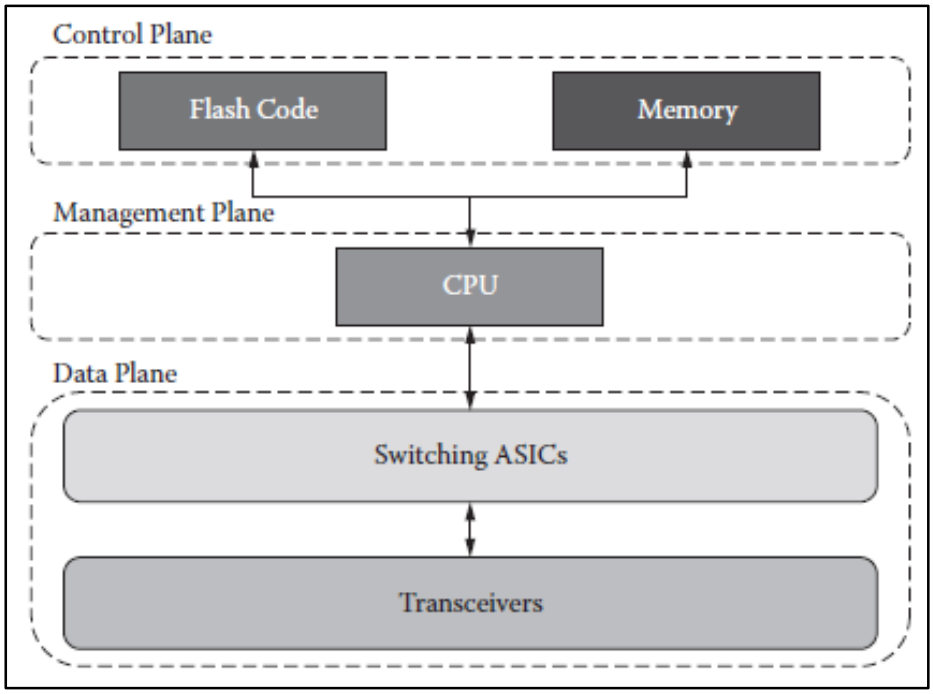


Figura 1. Tres planos de un router tradicional
Fuente: (Morreale & Anderson, 2015).

Visto desde el punto vista de los planos de manera separada, el programa o software de control se puede ejecutar en uno o más servidores y el NOS o Sistema Operativo de Red puede ejecutarse en otro conjunto diferente de servidores. El NOS observara y controlará el plano de datos, aunque este no sea parte de él.

Uno de los impactos potenciales de la implementación de SDN es la posibilidad de que el diseñador de redes de una empresa pueda adquirir el plano de control de terceros proveedores, haciéndolo independientemente de los proveedores que proporcionan el hardware de conmutación. Esto es permisible debido a que la inteligencia de red se elimina de los conmutadores y ahora reside en las capas SDN. Los switches se vuelven únicamente hardware de producto.

Otro de los cambios que suceden tras la implementación de una SDN es como se realizan las pruebas de red. Actualmente la única manera de probar una red es construir un entorno de prueba

con equipamiento de lasmismas características del que se planea disponer en la red real; así usando scripts prediseñados se verifica el funcionamiento y que tan bien elaborado está el perfil de red.

En SDN el enfoque cambia totalmente ya que todo el hardware se halla anclado a una interfaz común y el plano de control ahora existe en el software. Gracias a esto es posible realizar pruebas de unidad de hardware de manera segura, se puede conseguir simulaciones a gran escala del plano de control y probar un diseño antes de ponerlo en producción. (Morreale & Anderson, 2015)

2.1.3.2. Control Centralizado

Basándose en la idea de separación del plano de reenvío y control la siguiente característica fundamental de las SDN es la simplificación de los dispositivos, que pasan a ser gestionados desde un elemento central denominado software de control o manager. Así en lugar de poseer miles de líneas de software de control en el dispositivo para que se comporte de manera autónoma, ese software se elimina y se coloca un controlador centralizado; que será capaz de administrar la red mediante políticas de nivel superior enviando únicamente instrucciones primitivas a los dispositivos simplificados evitando a estos la toma de decisiones sobre el manejo de paquetes.

2.1.3.3. Automatización y virtualización de la red

La virtualización se relaciona con mayor frecuencia a los términos computación y almacenamiento, pero en realidad ambos son los que generan demanda de virtualización de las redes particularmente en los centros de datos. La virtualización de la red proporciona un servicio que se encuentra desacoplado del hardware físico que ofrece un conjunto de características idéntico al comportamiento de su contraparte física.

Un acercamiento temprano a la virtualización era el uso de VLAN que permitieron la coexistencia de dos redes físicas en la misma capa en total aislamiento entre sí. A pesar de que el

concepto es muy sólido el aprovisionamiento de VLAN's no es dinámico y escala únicamente a una topología de capa dos. En Capa tres la escala de túneles es mejor permitiendo que los sistemas evolucionen en el uso de VLAN y túneles para proporcionar soluciones de virtualización de red.

Uno de los esfuerzos comerciales más exitosos fue el de Nicira que hoy es parte de la empresa VMware. Inicialmente Nicira afirmó que había siete propiedades de virtualización de red:

- Independencia del Hardware de Red
- Fiel repoducción del modelo de servicio de red física
- Siguiendo un modelo operativo de virtualización computacional
- Compatibilidad con cualquier plataforma de hipervisor
- Aislamiento seguro entre las redes virtuales, las redes físicas y el plano de control
- Rendimiento y escala de la nube
- Provisión y control programático de la red

Varias de las características del modelo de Nicira aciertan en los parámetros que SDN expone actualmente, esto promete un mecanismo para automatizar la red y extraer el hardware físico bajo la red definida por software. Debido a que la virtualización se ha realizado en su mayoría en centros de datos y que SDN es simplemente abstraer el plano de control y el plano de datos, el termino virtualización de red se ha vuelto sinónimo de SDN. (Goransson, Black, & Davy, 2014)

2.2. ARQUITECTURA Y ELEMENTOS DE SDN

Los dispositivos que interconectan una red se encuentran compuestos por un plano de datos encargado de transmitir cada paquete a través de la red y un plano de control que determina cual es el mejor camino para que la información llegue a su destino. Debido a que SDN separa o abstrae estos dos planos es posible disponer de una interfaz abierta para ofrecer una solución a la demanda y requerimientos de cada punto de conexión.

La representación general de las SDN se realiza mediante un diagrama que muestra la relación de los dos planos y sus tres capas embebidas que son aplicación, control e infraestructura de las cuales las dos primeras se hallan gestionadas en el plano de control y la ultima es parte del plano de datos. El protocolo OpenFlow hace las veces de conector entre la capa de infraestructura y los dos superiores mediante su comunicación con el NOS.

SDN es modelado como un conjunto de relaciones cliente-servidor con un núcleo constituido por el controlador de SDN como indica la Figura 2. Las facilidades que ofrece este modelo están orientadas tanto a clientes como a proveedores de servicios, permitiendo una mejor gestión, distribución del procesamiento y orquestación de recursos para el cumplimiento del servicio. (Open Networking Fundation, 2017)

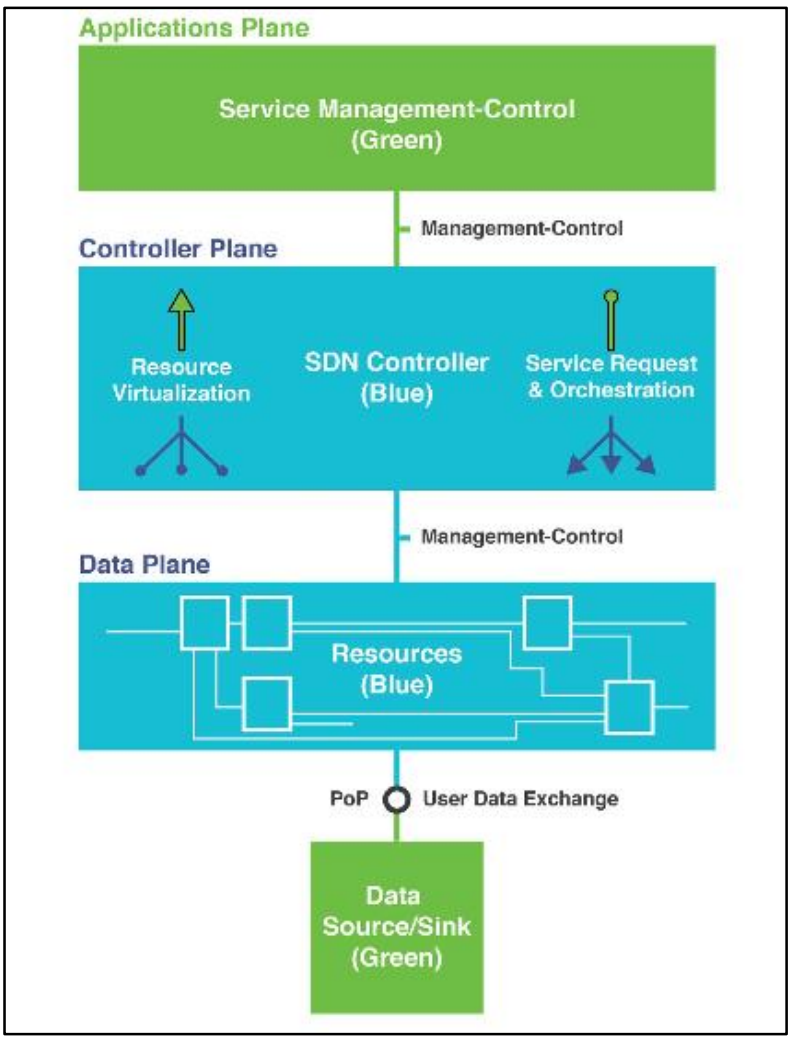


Figura 2. Modelo Básico de SDN
Fuente: (Open Networking Fundation, 2017).

EL propósito de SDN es reducir costos y mejorar la experiencia del usuario al automatizar los servicios de red desde el usuario hasta los elementos de red. Los principios que promueven esto incluyen el desacoplamiento del control de tráfico, procesamiento y reenvío, centralización del control y la capacidad de los clientes y aplicaciones para interactuar directamente con el control de red. La Figura 3 detalla los componentes de la arquitectura de un ambiente SDN relacionando el control, la administración y los requerimientos del usuario.

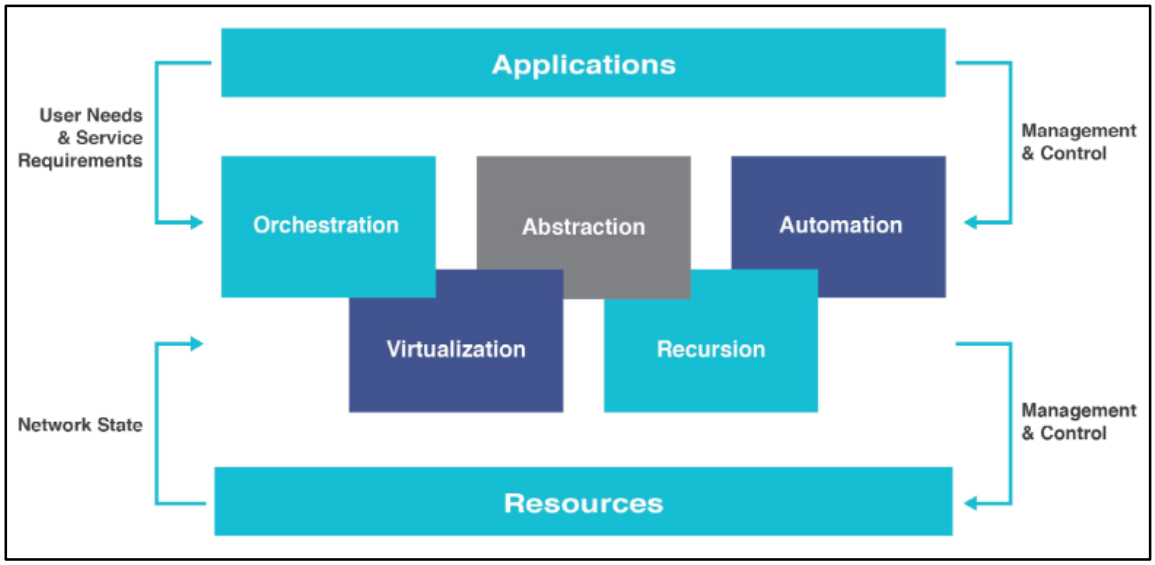


Figura 3. Arquitectura de una Red Definida por Software
Fuente: (Open Networking Fundation, 2017).

Un ambiente SDN está constituido por varios elementos que en simbiosis conforman la arquitectura de red. En primer lugar, se encuentran los planos de control y de datos separados tácitamente, estos planos son manejados por el controlador de red que es el encargado de gestionar sus políticas a través del Network Operación System que a su vez se comunica con el hardware de red mediante el protocolo estándar de SDN denominado OpenFlow, permitiendo generar el denominado ambiente virtual de red.

2.2.1. PLANO DE CONTROL DE RED

El plano de control establece el grupo de datos local usado para crear las entradas de la tabla de reenvío, las cuales son utilizadas por el plano de datos para reenviar el tráfico entre los puertos de acceso de un dispositivo de red; este conjunto de datos que representan una topología almacenada se denomina Tabla de Enrutamiento o Base de Información de Enrutamiento(RIB).

El plano de control puede mantener múltiples instancias en diferentes puntos de la red y aun así la RIB se mantendrá estática o libre de bucles de enrutamiento durante el intercambio de

información debido a que siempre se espera la estabilización de las rutas. La FIB o Base de Información de Reenvió se refleja constantemente entre los planos de control y de datos de un dispositivo de red típico. La FIB se programa una vez que la RIB se vuelve completamente estable. (Nadeau & Gray, 2013)

EL plano de control de red está diseñado para calcular el estado de envío en base a tres restricciones especiales:

- Tiene que ser coherente con el software. El ASIC debe estar disponible en la interfaz CLI para saber q es lo que hacen el hardware y el software del conmutador.
- Debe estar basado en la topología de red completa.
- El estado de reenvió debe difundirse en cada uno de los enrutadores de la red.

Cada vez que se diseña un nuevo protocolo se revisan estos tres problemas y a partir de ellos se crea una solución nueva. Este enfoque no es razonable respecto a las (NGN) Redes de Nueva Generación. (Morreale & Anderson, 2015)

La Figura 4 muestra la diferente arquitectura que tendría un enrutador en SDN. La funcionalidad del plano de datos seria lo único que proporcione el hardware del dispositivo. El plano de control y el de gestión serían proporcionados por una aplicación de software que se ejecuta en una plataforma externa al dispositivo conectada al enrutador a través de una conexión o enlace seguro.

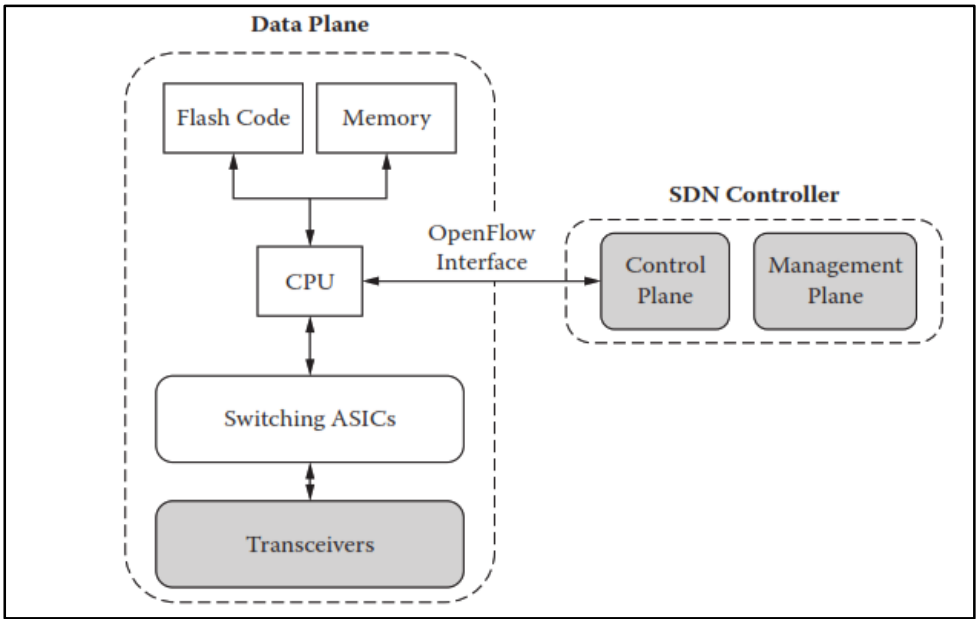


Figura 4. Componentes de un router en SDN
Fuente: (Morreale & Anderson, 2015).

El plano de control define ciertas funciones específicas dentro de su papel en la red. El más básico que es un modelo de reenvío general que oculta información sensible del conmutador de red. Luego está la función para detectar el estado actual de la red, usada para tomar decisiones basadas en la complejidad de la red. Y finalmente existe una función de configuración de red que evita tener que configurar manualmente cada router de la red, esta simplifica la configuración mediante el cálculo de configuración de manera automática entre dispositivos físicos.

2.2.1.1. Función de Reenvío (Forwarding Function)

La función de reenvío debe implementarse independientemente de cómo se implementa el conmutador de red. Esto significa que el conmutador de red debería poder utilizar cualquier conjunto de ASIC con distintos grados de capacidad y esto no debería tener ningún efecto en la función de reenvío. Además, el software que este corriendo sobre el conmutador es indiferente del proveedor así que no debería causar impacto alguno. (Morreale & Anderson, 2015)

Dentro del mundo de SDN algunos de los fabricantes han realizado propuestas para implementar la función de reenvió, pero a base del desarrollo colaborativo OpenFlow se convirtió en el protocolo estándar o de facto para la interacción con los paquetes de datos. Se introduce el termino flow o flujo para definir a las entradas de paquetes en las API's que soportan OpenFlow, haciendo que este último se vuelva simplemente un lenguaje que los switches deben entender. De manera general la implementación de la función de reenvió a base de OpenFlow está basada en decisiones que incluyen el matching o coincidencia de flujos para permitir, bloquear o re direccionar el tráfico. (Nadeau & Gray, 2013)

2.2.1.2. Función de Estado de Red (Network State Function)

La función de estado de red debe presentar una vista de red global, algo parecido a un gráfico de objetos y enlaces que tiene información asociada. Una vez que se crea este grafico de la red, el software de control puede tomar decisiones sobre qué hacer en función del grafico de red. Si el acceso al grafico es a través de una API, entonces los elementos de la red reales se controlan desde la misma. (Morreale & Anderson, 2015)

La funcionalidad de la vista de red global se puede implementar como parte del sistema operativo de red, y para incrementar la confiabilidad se puede replicar en distintos servidores en diferentes puntos de la red. Para mantener la información actual de la red debe existir flujo bidireccional entre el sistema operativo de red y los servidores de red como muestra la Figura 5, el flujo de información constante permitirá que la información se actualice mostrando el estado real de los switches en todo momento. Este tipo de flujos puede servir también para que los switch intercambien información de control que puede ser usada para actualizar la vista de la red de manera individual.

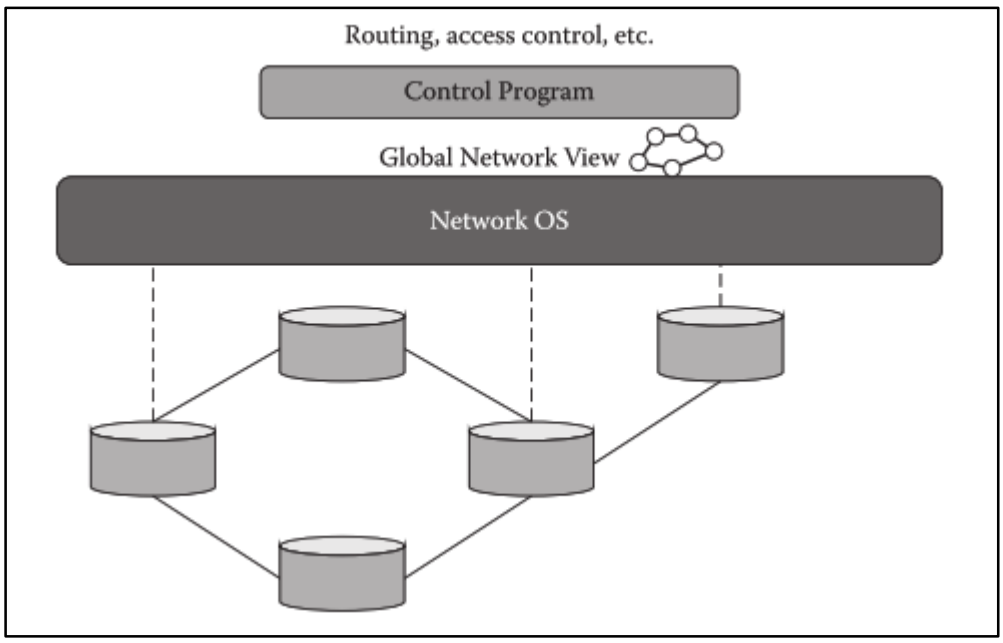


Figura 5. SDN con vista global de la red y programa de control.
Fuente: (Morreale & Anderson, 2015).

2.2.1.3. Función de Configuración (Configuration Function)

A través de la función configuración los nodos son capaces de saber cuál es la ruta para el envío de un paquete ya que por sí mismos no son capaces de hacerlo, el encargado de establecer las políticas es el programa o software de control instalado en el dispositivo o servidor que hace las veces de controlador de red. La función de configuración es capaz de interactuar con el grafico global de red para determinar expresamente cuales son las restricciones y los permisos de cada nodo en la red. Esencialmente estos detalles permiten que el software de control pueda aplicar reglas o instrucciones a los flujos como QoS, ACL y TE en diversos niveles.

La función de configuración, así como en el mundo de la programación y los compiladores interpreta instrucciones escritas en lenguajes de alto nivel y los traduce a lenguaje de máquina para que así sea más sencillo para el administrador escribir un script de configuración. La única desventaja es que el programa de control no discrimina entre información crítica e información

innecesaria, por lo que aparece una nueva capa en el modelo de SDN denominada Capa de Virtualización que es la encargada de crear un modelo abstracto de la red en el software de control que muestra solo la información importante junto a una vista simple de la red. La capa de virtualización puede tomar decisiones de control de acceso del programa de control y difundirlas en la vista global. (Morreale & Anderson, 2015)

2.2.1.4. Aplicaciones sobre hardware de SDN

El servidor encargado de ejecutar el plano de control puede ser capaz también de correr una serie de aplicaciones de red que son comunes en los routers tradicionales de las redes actuales. Entre este grupo de aplicaciones podemos incluir el reenvío de paquetes, la creación de redes privadas virtuales (VPN's), gestión de seguridad, control de ancho de banda, virtualización de red, balanceo de carga entre otros.

La política general para el establecimiento de una topología es ubicar los routers y los firewalls en el borde de la red en donde el límite esta dictado por el control de acceso. Las empresas de hoy en día están empezando a trasladar sus redes corporativas del dominio físico al dominio del cloud computing, usando la topología actualmente implementada para replicar las políticas al entorno basado en cloud.

El verdadero poder de las SDN radica en la posibilidad de que el operador o administrador de red pueda establecer la topología virtual de red en su red empresarial, ignorando el diseño físico y reemplazando las políticas basadas en hardware por políticas basadas en software, permitiendo la modificación dinámica de la estructura de la red y el aprovisionamiento dinámico de los componentes y dispositivos de red basándose en las tres funciones en las que consiste el plano de control. (Goransson, Black, & Davy, 2014)

2.2.2. PLANO DE DATOS

El plano de datos es el encargado de manejar los paquetes entrantes a través de una serie de enlaces independientemente de la tecnología de la interfaz física, durante el proceso de recolección se realiza una serie de comprobaciones de consistencia en los paquetes para que así puedan ser enviados luego al matching con las entradas de la tabla de reenvío programadas por el plano de control.

La mayoría de veces la recolección y reenvío se refiere a la búsqueda de la ruta más rápida para el procesamiento de paquetes basado en la FIB pre programada, pero cuando un paquete no tiene coincidencia con ninguna de las reglas y se detecta un destino desconocido los paquetes son enviados al procesador de ruta donde el plano de control realiza un proceso adicional buscando una ruta posible en la RIB. (Nadeau & Gray, 2013)

Respecto a las decisiones de reenvío, el plano de datos incorpora una serie de características relacionadas con el reenvío de tráfico; en algunos sistemas suelen encontrarse en tablas discretas mientras que en otras se comportan como extensiones de las tablas de reenvío lo que provoca que las entradas se vuelvan más extensas. Dependiendo del diseño las características relacionadas con el reenvío se implementan en un orden específico de acuerdo a la exclusividad de sus funciones como indica la Figura 6.

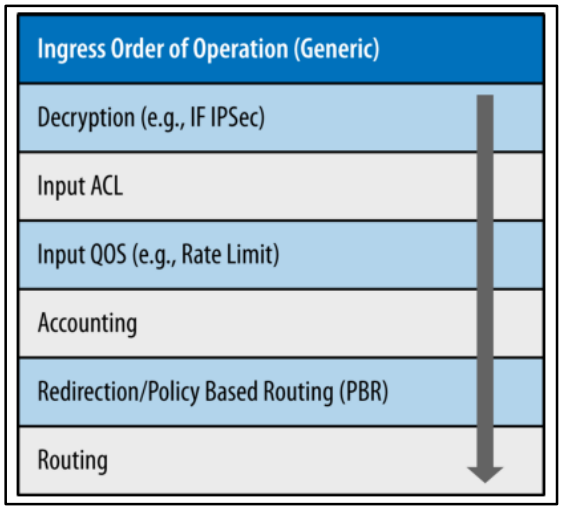


Figura 6. Ejemplo genérico del ingreso de aplicaciones características
Fuente: (Nadeau & Gray, 2013).

2.2.3. MOVIMIENTO DE INFORMACIÓN ENTRE PLANOS

Los sistemas multi slot/multi card basados en chasis más robustos en actualidad están imitando el comportamiento de los mecanismos de control lógicamente centralizados, pero físicamente distribuidos de SDN. Es decir que los sistemas que trabajan con tarjetas de función simulan un plano de control externalizado, así si este reside en una tarjeta procesadora y los planos de datos se encuentran en otras tarjetas en líneas independientes existirán comportamientos en torno a la comunicación entre estos elementos del sistema para que se logre estabilidad, resistencia y tolerancia a fallas.

Los mecanismos para detección de errores en las tablas de reenvío son un punto clave para tener un sistema consistente, de esta manera se asegura que las versiones de las tablas se encuentran actualizadas valiéndose del hash o las firmas generadas a partir del contenido. A la vez estos mismos mecanismos aseguran que las versiones distribuidas de software de las tablas se encuentren correctamente programadas y sincronizadas.

En ciertas ocasiones durante el ciclo de comunicación del plano de control con el plano de datos se generan ciertos problemas de agujero negro, resultantes de algoritmos de sincronización y/o actualización de tabla ineficientes en los sistemas encargados de crear las entradas de reenvío combinando información de tablas separadas como en el caso cuando la dirección de hardware de un siguiente salto a un destino no está poblada en una tabla de adyacencia pero una ruta que utiliza ese siguiente salto llena la tabla de rutas, lo que lleva a una entrada de reenvío "no resuelta".

En el lado izquierdo de la Figura 7 se muestra un enrutador/conmutador multi slot que realiza una búsqueda en dos etapas. Si observamos el enlace A-B en el lado derecho la búsqueda de ingreso de FIB resultante en la tarjeta 1 cambia de la tarjeta 3 a la tarjeta 2. Si la actualización de la tarjeta 2 ocurre después de 1 y 3, la búsqueda secundaria fallará. De manera similar, en un entorno SDN, si el túnel que conecta A y B cambia de la interfaz 3 a la interfaz 2 en estos sistemas, entonces la asignación de los flujos de 1-3 a 1-2 en estos elementos tiene que ser sincronizado por la aplicación en el controlador SDN indicado como CP en la figura. Estas técnicas u optimizaciones de mitigación se llevan a cabo con el propósito de resaltar la importancia de la coherencia en el contexto de centralizar el plano de control.

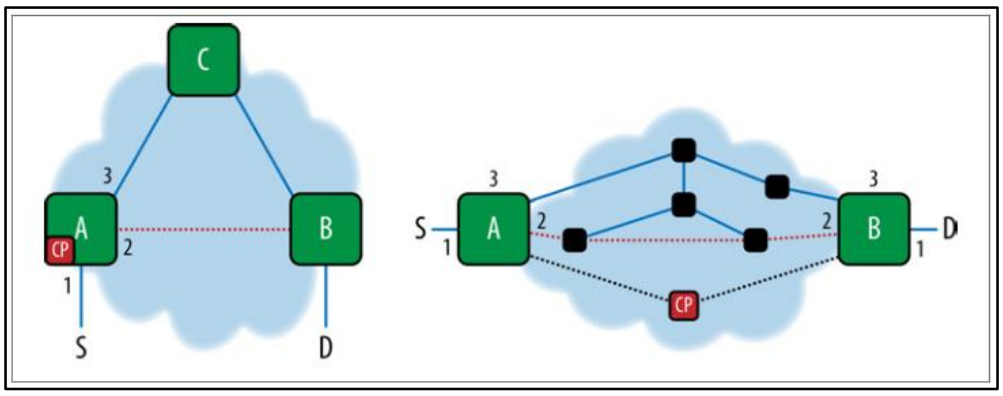


Figura 7. Pérdida Asincrónica en dos etapas
Fuente: (Nadeau & Gray, 2013).

2.2.4. DISPOSITIVOS DE RED DEFINIDOS POR SOFTWARE

Una red de datos comprende todo el hardware y el software que permite que se lleve a cabo la comunicación entre dos extremos, con el tiempo la demanda de canales de comunicación digitales se ha vuelto cada vez más alta y por esta razón la parte que más modificaciones ha tenido que sufrir es la del hardware de red que se vuelve obsoleto mayor rapidez.

Con el desarrollo acelerado de la tecnología se ha logrado mitigar en gran manera la obsolescencia de gran parte del hardware de red usando versiones con mejoras o instalando hardware de expansión, y aun así se sufre de un alto grado de insuficiencia para cubrir las demandas de los usuarios mediante el aprovisionamiento de equipos físicos. De aquí nace la necesidad de implementar nuevas estrategias y nuevas tecnologías de red que permitan disminuir la cantidad de hardware necesario para el despliegue de entornos de red especialmente en el lado de los consumidores.

El termino más común para la solución alternativa al hardware es la virtualización, que en un inicio supone el uso de componentes generales de hardware embebidos en equipos más potentes. A partir del éxito de la virtualización de la memoria, almacenamiento y posteriormente la virtualización completa de servidores, se han planteado desafíos más grandes cada vez que poco a poco han generado las soluciones más efectivas para los problemas más comunes en el área de la informática. Es así que en los últimos años se han desarrollado una serie de dispositivos virtuales que son capaces de emular el comportamiento del equipamiento de red físico.

Un dispositivo de red virtual es un elemento que se encuentra definido en un dominio conectado a un conmutador virtual, administrado por un controlador de red virtual y conectado a una red virtual a través de un hipervisor usando canales de dominio lógico (LDC) como se observa en la

Figura 8. Los dispositivos de red virtual representan interfaces de red con el nombre vnet*n* y son capaces de funcionar como cualquier interfaz de red común. (ORACLE, 2011)

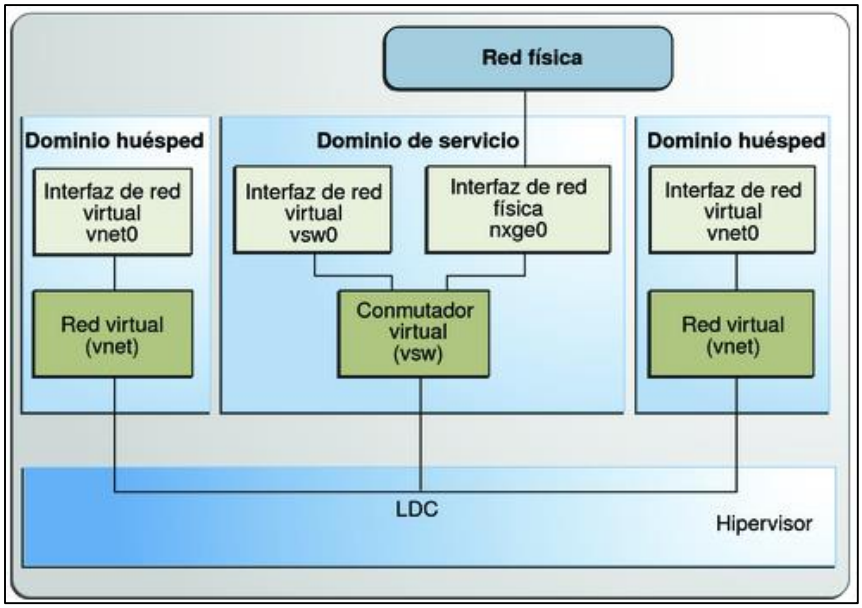


Figura 8. Canales ente redes virtuales
Fuente: (ORACLE, 2011).

Como se describe anteriormente existen también conmutadores y enrutadores de red virtuales que cumplen funciones similares a las sus contrapartes físicas, estos dispositivos presentan también ciertas funciones adicionales que permiten tener mayor control sobre el tráfico y flujo de paquetes mediante su activación a manera de módulos basados en software incluidos en el sistema operativo de red.

2.2.5. HARDWARE PARA RED DEFINIDA POR SOFTWARE

Un dispositivo SDN o definido por software es una unidad compuesta por una Interfaz de Programación de Aplicaciones para comunicarse con el controlador, una capa de abstracción y una función de procesamiento de paquetes. Para el caso de los conmutadores virtuales, la función de procesamiento de paquetes se encuentra hasta debajo de los niveles de ejecución como indica en

el extremo izquierdo de la Figura 9. Si el conmutador es físico, la función de procesamiento de paquetes se halla incorporada en el hardware para la lógica como se muestra en el lado derecho de la Figura 9.

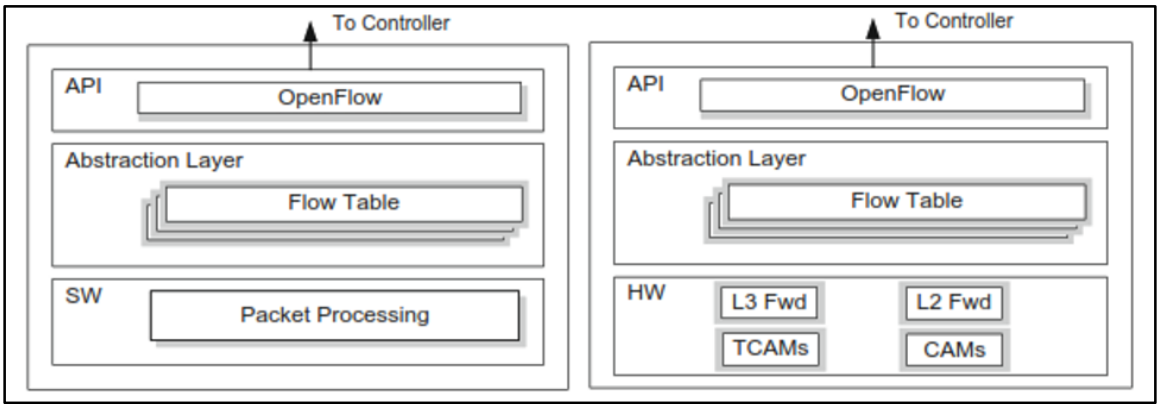


Figura 9. Anatomía de un switch SDN Virtual y un switch SDN físico.
Fuente: (Goransson, Black, & Davy, 2014).

La capa de abstracción puede incorporar una o más tablas de flujo y la lógica de procesamiento de paquetes consistirá en mecanismos para toma de decisiones basadas en los resultados de la evaluación de los paquetes entrantes y la búsqueda de la coincidencia o match de mayor prioridad. Una vez que se encuentra una coincidencia, el paquete entrante es procesado localmente a menos que se reenvíe explícitamente en el controlador. En el caso de no existir coincidencia el paquete puede ser copiado en el controlador para su posterior procesamiento. Este proceso se conoce como paquete consumido por el controlador. En el caso de un conmutador físico todos estos mecanismos se ejecutan a nivel de hardware específico para este trabajo. (Goransson, Black, & Davy, 2014)

La lógica real de reenvío de paquetes de los enrutadores con el tiempo ha tenido que emigrar al hardware para acoplarse a los switches que necesitaban procesar paquetes que llegaban a velocidades de línea cada vez mayores. Recientemente en los centros de datos ha surgido un rol que propone un cambio hacia el software puro. Los conmutadores se implementan como

aplicaciones de software que generalmente se ejecutan de manera adyacente al hipervisor en un rack del centro de datos. Al igual que una máquina virtual, el switch virtual puede ser instanciado o movido bajo control de software y funciona normalmente como un conmutador virtual con un conjunto de otros conmutadores virtuales para constituir una red virtual. (CISCO, 2016)

2.2.6. VIRTUALIZACIÓN DE FUNCIONES DE RED (NFV)

Dentro del ámbito de las SDN existe un término muy importante llamado NFV o Virtualización de Funciones de Red, en torno a él existen confusiones en la industria sobre si es o no una forma de SDN. NFV es la implementación de la funcionalidad de los dispositivos y aplicaciones de red, pero basadas en software. Esto significa implementar un componente general de la red como un balanceador de carga o un IDS a base de software. Por esta razón es posible llegar a virtualizar enrutadores y conmutadores, gracias a los avances en el hardware de servidor de uso general.

Aunque los avances de hardware se han implementado inicialmente para admitir la gran cantidad de máquinas virtuales y de conmutadores necesarios para realizar la virtualización de red, tienen la ventaja de que son dispositivos completamente programables y, por lo tanto, se pueden reprogramar para que sean muchos tipos diferentes de dispositivos de red. Esto es particularmente importante en el centro de datos del proveedor de servicios, donde la flexibilidad para reconfigurar dinámicamente un firewall en un IDS en función de la demanda del inquilino es de enorme valor. (Goransson, Black, & Davy, 2014)

En la orquestación de un centro de datos es posible ofrecer servicios que incluyen el uso de dispositivos virtuales como firewalls. En general, el uso de estos dispositivos se puede orquestar de manera que permita el recorrido simple de un canal de operaciones donde las interfaces lógicas que representan tanto un ingreso como una salida de la red de inquilinos crean un patrón de flujo

de tráfico simple. Este es el núcleo de la virtualización de funciones de red (NFV) y esas funciones se encadenan usando un concepto denominado encadenamiento de servicio.

El meta concepto alrededor de NFV es que en la definición del servicio para todos los inquilinos como se muestra en la Figura 10 es una cadena fundamental, en este caso ADC-Firewall para todo ingreso de tráfico en internet. En realidad, se trata de dos, en el caso de VPN las acciones del dispositivo son hacia adelante, modificar y reenviar o descartar. No se requiere lógica adiciona, no se pasan metadatos de un elemento de servicio a otro y no hay una lógica de derivación significativa en la cadena.

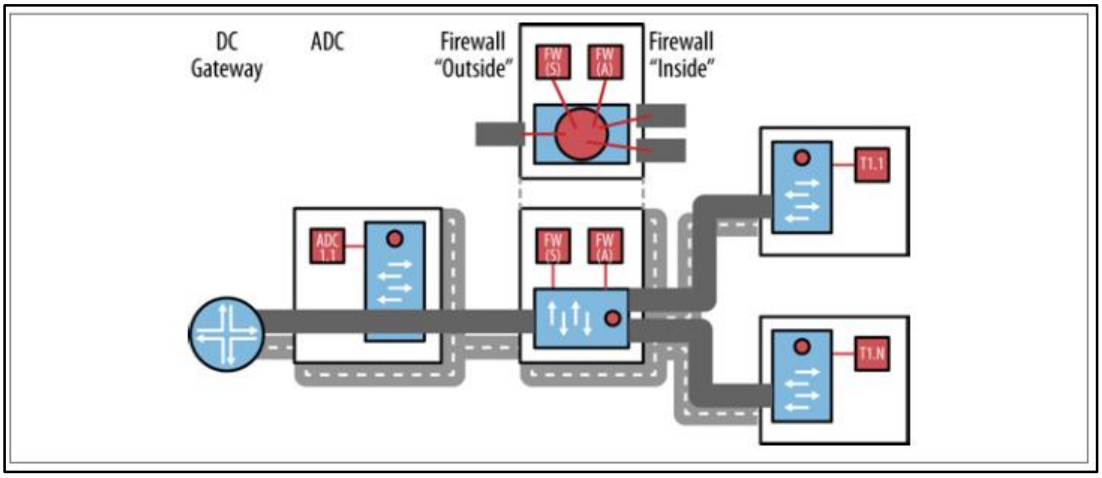


Figura 10. Cadena de servicio en un centro de datos
Fuente: (Nadeau & Gray, 2013).

Las cadenas de servicio generalmente se construyen por alguna entidad de control u orquestación, es decir, un controlador SDN. Esta entidad es responsable del aprovisionamiento de los servicios, y luego el encadenamiento de los mismos. La configuración real por habitante de los servicios, como los cortafuegos y la puerta de enlace, puede variar, pero debe ser mantenida por el controlador NFV u orquestador. La transparencia del esfuerzo de aprovisionamiento a este punto

en el tiempo puede depender del proveedor de orquestación seleccionado y del producto desplegado.

En el caso de que los elementos sean del mismo proveedor existe una buena posibilidad de que la implementación de flujos a través del aprovisionamiento sea posible. Si no lo son, se puede mantener cierto grado de transparencia a través de un intermediario de nivel superior u Operations Support System (OSS) que interactúa tanto con el controlador SDN como con el proveedor de cortafuegos Element Management System (EMS) o entidad de aprovisionamiento. Lo mismo es realizable sobre cualquier aplicativo / cadena de servicio. (Nadeau & Gray, 2013)

La razón de la confusión sobre si NFV es una forma de SDN es que las dos ideas son complementarias y superpuestas. SDN se puede usar muy naturalmente para implementar ciertas partes de NFV. Sin embargo, el NFV puede ser implementado por tecnologías que no sean SDN, del mismo modo que SDN puede usarse para muchos fines no relacionados con el NFV. (Goransson, Black, & Davy, 2014)

2.2.7. VINCULO ENTRE NFV Y SDN

Es difícil conseguir que los equipos de nivel empresarial tengan un ciclo de depreciación largo, teniendo en cuenta que estos son encargados de procesar los grandes volúmenes de información que se generan día a día y que soportan cargas bastante altas si se trata de manejar altos índices de tráfico y flujos densos de paquetes.

Tanto las Redes Definidas por Software (SDN) como la Virtualización de Funciones de Red (NFV) proponen desarrollos revolucionarios, y el éxito de cualquiera de ellos al cambiar la red depende de la armonía entre las tecnologías utilizadas y su compatibilidad entre sí. Justo donde se

ubique el nivel de armonía de estos elementos será en donde se empiece a trazar la ruta a las redes del futuro.

SDN se ha propuesto para abordar las necesidades de servicios de red a través de la virtualización de funciones de red (NFV). La idea de NFV es mover la funcionalidad del servicio (balanceadores de carga, firewalls, entre otros) de dispositivos especializados e implementarlo como software que se ejecuta en plataformas de servidor comunes. Esto ha sido posible gracias a los avances en la tecnología de interfaz de servidor y de red.

Las aplicaciones que antes requerían hardware especializado para CPU y NIC ahora pueden ser ejecutadas por servidores estándar de la industria. Los argumentos a favor de este cambio incluyen el menor costo de los servidores en comparación con estos dispositivos especializados, así como la capacidad de actualizar fácilmente el hardware cuando sea necesario. (Goransson, Black, & Davy, 2014)

Mientras las SDN pueden ser utilizadas sin NFV y viceversa, el poder real, sobre todo en lo relacionado con el Cloud Computing, viene cuando se utilizan en juntos. La combinación de estas dos tecnologías y sus mecanismos permiten ofrecer aprovisionamiento automático y centralizar la gestión y el control en ambientes IaaS.

SDN y NFV se pueden usar con cualquier modelo de implementación, pero en el modelo de nube publica es importante únicamente para el proveedor de servicios, no para la mayoría de las empresas, mientras que, en los modelos privados e híbridos, los clientes deben tener en cuenta como asignan los recursos de la empresa. (Interoute Iberia, 2014)

2.3. OPENFLOW

OpenFlow es el protocolo que estructura la comunicación entre los planos de control y de datos en los dispositivos de red que lo admiten. Ha sido diseñado para proporcionar una aplicación externa con acceso al plano de reenvío de un conmutador de red o en su defecto de un enrutador. El acceso a esta parte del enrutador se puede obtener a través de la red, lo que permite que el programa de control no tenga que estar ubicado cerca del conmutador.

La especificación OpenFlow ha estado evolucionando durante varios años. La organización de Internet no lucrativa OpenFlow.org se creó en el año 2008 como base para promover y apoyar OpenFlow. La primera versión, la 1.0.0, apareció el 31 de diciembre de 2009, aunque existían numerosas versiones previas eran solo experimentales. En este punto y continuando hasta la versión 1.1.0, el desarrollo y la administración de la especificación se realizaron bajo auspicios de OpenFlow.org. El 21 de marzo de 2011, se crea la Open Networking Foundation (ONF) con el propósito expreso de acelerar la entrega y comercialización de SDN.

Existen varios componentes de SDN que ofrecen soluciones SDN que no están basadas en OpenFlow. Sin embargo, OpenFlow sigue siendo el núcleo de la visión de SDN para el futuro. La ONF se ha convertido en la entidad responsable de la evolución de la especificación de OpenFlow. Comenzando después del lanzamiento de la versión 1.1, se han publicado y administrado las revisiones de la especificación.

En la Figura 11, se muestran los componentes clave del modelo OpenFlow que últimamente se han convertido el al menos parte de la definición común de SDN principalmente:

- Separación de los planos de control y de datos.

- Uso del protocolo estandarizado entre el controlador y un agente en la red elemento para el estado de creación de instancias.

- Proporcionar programabilidad de red desde una vista centralizada a través de una API moderna y extensible.

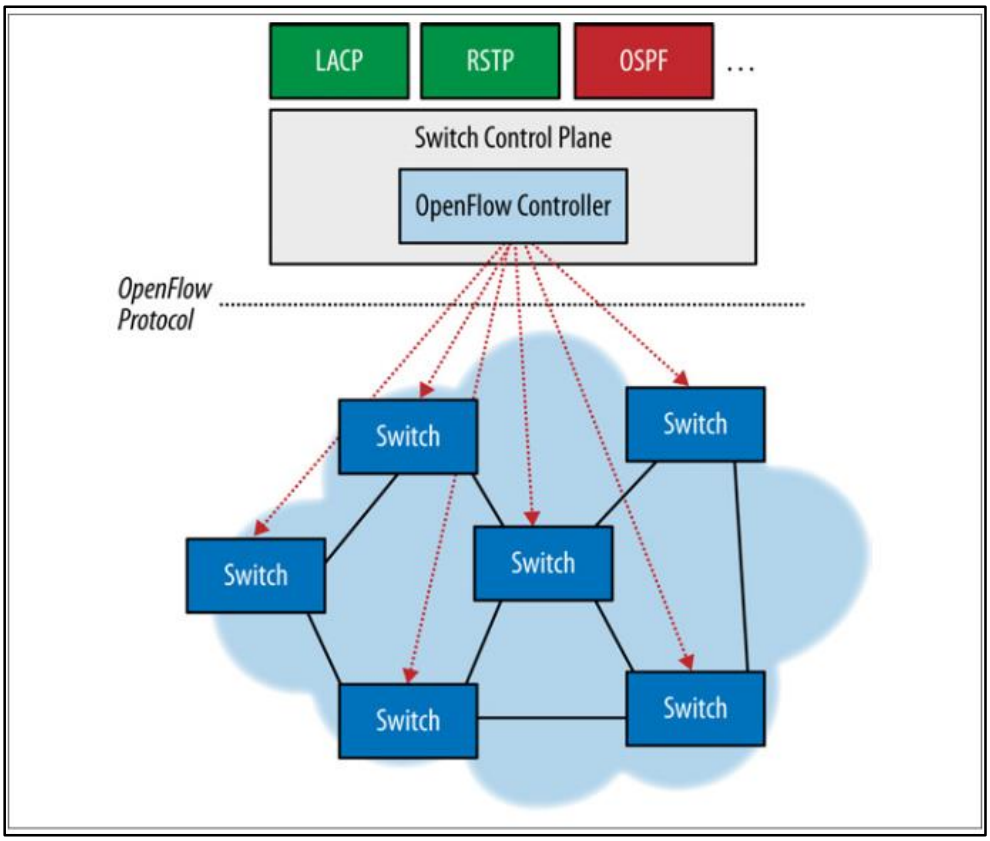


Figura 11. Arquitectura de OpenFlow
Fuente: (Nadeau & Gray, 2013).

2.3.1. EVOLUCIÓN DE OPENFLOW

Debido a la trascendencia que se esperaba con el cambio a SDN, las organizaciones participantes del proyecto OpenFlow fomentaron el uso de esta especificación en múltiples entornos de prueba para así poder establecer estándares y lograr la implantación de las SDN más allá de las academias y grupos de desarrollo experimentales. A partir de este impulso OpenFlow

logro destacar y empezó a ser incluido en una gran variedad de firmwares y sistemas operativos de red de las organizaciones miembros del proyecto, consiguiendo un grado de evolución bastante alto en poco tiempo. Las versiones funcionales existentes de la especificación OpenFlow van desde la 1.0 hasta la 1.5, de las cuales la 1.4 y la 1.5 aún se encuentran en desarrollo y pretenden incluir soporte para las redes de fibra óptica de manera nativa, así como lotes de instrucciones extendidos para las tablas de flujo.

2.3.1.1. OpenFlow v1.0

OpenFlow versión 1.0 fue lanzado el 31 de diciembre de 2009. A los efectos de este trabajo, tratamos OpenFlow 1.0 como el lanzamiento inicial de OpenFlow. De hecho, años de trabajo y puntos múltiples precedieron al lanzamiento de la primera versión publica de OpenFlow, pero incluyendo a esto el proceso incremental en el lanzamiento inicial como si este hubiese ocurrido atómicamente.

La especificación de OpenFlow define el concepto de un puerto OpenFlow correspondiente a un puerto físico que durante años ha admitido múltiples colas, generalmente atendidas por algoritmos de programación que permiten el aprovisionamiento de distintos niveles de calidad de servicio (QoS) para diferentes tipos de paquetes. OpenFlow adopta este concepto y permite mapear un flujo a una cola ya definida en un puerto de salida. Por lo tanto, si observamos la Figura 12, la salida de un paquete en el puerto nombrado N puede incluir la especificación de la cola en el puerto N en el que debe colocarse el paquete. Se aprecia también que el recuadro de acciones puso en cola específicamente el paquete que se está procesando en la cola 1 en el puerto N.

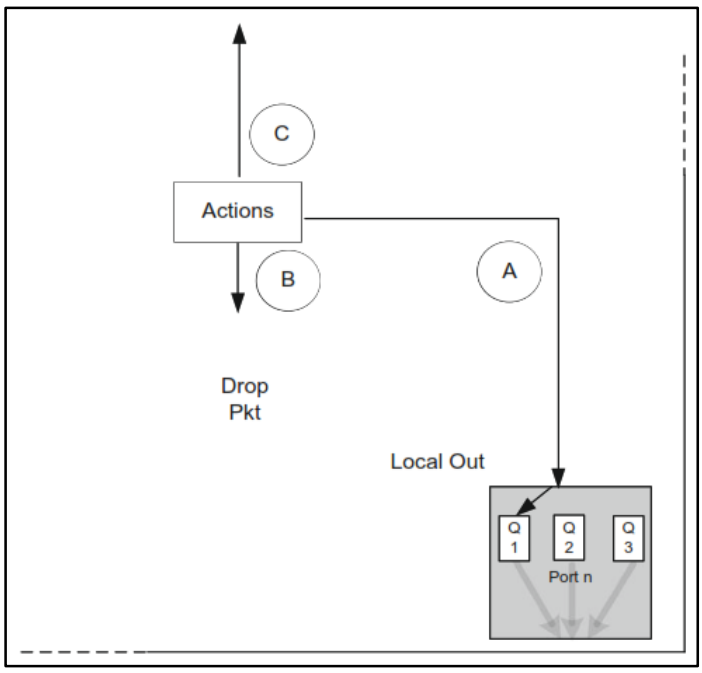
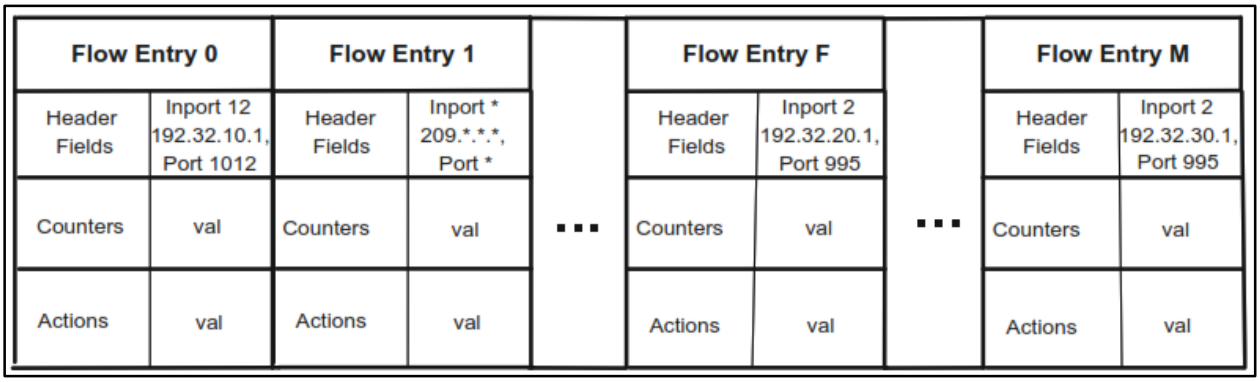


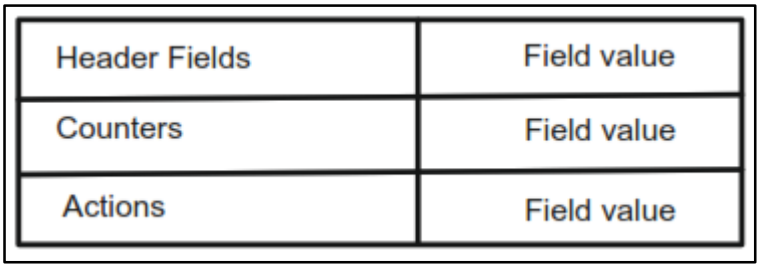
Figura 12. Soporte de OpenFlow para multiples colas por puerto.
Fuente: (Goransson, Black, & Davy, 2014).

La tabla de flujo se encuentra en el núcleo de la definición de un switch OpenFlow. En la figura 13 se representa una tabla de flujo genérica la cual consta de una serie de entradas de flujo que se muestran en la figura 14. Una entrada de flujo consiste en campos de encabezado, contadores y acciones asociadas con esta entrada. Los campos de encabezado se utilizan como criterios de coincidencia para determinar si un paquete entrante coincide con esta entrada. Si existe una coincidencia, el paquete pertenece a este flujo. Los contadores se utilizan para rastrear las estadísticas relativas a este flujo, como cuantos paquetes se han reenviado eliminado para este flujo. Los campos de acciones indican que debe hacer el cambio con un paquete que coincida con esta entrada. (OpenFlow, 2009)



Flow Entry 0		Flow Entry 1			Flow Entry F			Flow Entry M	
Header Fields	Inport 12 192.32.10.1, Port 1012	Header Fields	Inport * 209.*.*.*, Port *	...	Header Fields	Inport 2 192.32.20.1, Port 995	...	Header Fields	Inport 2 192.32.30.1, Port 995
Counters	val	Counters	val		Counters	val		Counters	val
Actions	val	Actions	val		Actions	val		Actions	val

Figura 13. Tabla de Flujo de OpenFlow v1.0
Fuente: (Goransson, Black, & Davy, 2014).



Header Fields	Field value
Counters	Field value
Actions	Field value

Figura 14. Entrada de Flujo Básica
Fuente: (Goransson, Black, & Davy, 2014).

Cuando un paquete llega al switch OpenFlow desde un puerto de entrada o en algunos casos desde el controlador, se compara con la tabla de flujo para determinar si hay una entrada de flujo coincidente. Los siguientes campos de coincidencia asociados con el paquete entrante se pueden usar para hacer coincidir contra entradas de flujo:

- Cambiar el puerto de entrada
- VLAN ID
- Prioridad de VLAN
- Dirección origen de Ethernet
- Dirección destino de Ethernet

- Tipo de trama Ethernet
- Direccion IP origen
- Dirección IP destino
- Protocolo IP
- Bits de tipo de servicio (ToS)
- Puerto origen TCP/UDP
- Puerto destino TCP/UDP

Los 12 campos se conocen como la colección básica de campos de coincidencia. Los campos de coincidencia de entrada de flujo pueden ser procesados con una máscara de bits, lo que significa que cualquier valor que coincida con los bits no enmascarados en los campos de coincidencia del paquete entrante será una coincidencia. (Goransson, Black, & Davy, 2014)

Las entradas de flujo se procesan en orden, y una vez que se encuentra una coincidencia, no se realizan más intentos con esa tabla de flujo. Por esta razón, es posible que existan múltiples entradas de flujo coincidentes para que un paquete esté presente en una tabla de flujo. Solo la primera entrada de flujo coincidente es significativa; los otros no se encontrarán, porque la coincidencia de paquetes se detiene en la primera.

La mensajería entre el controlador y el conmutador se transmite a través de un canal seguro. Este canal seguro se implementa a través de una conexión TLS inicial a través de TCP. Si el switch conoce la dirección IP del controlador, el conmutador comienza con el encabezado OpenFlow. Este encabezado especifica el número de versión de OpenFlow, el tipo de mensaje, la longitud del mensaje y la identificación de la transacción del mensaje. Los diversos tipos de mensajes en

OpenFlow v1.0 se enumeran en la Tabla 1. Los mensajes se dividen en tres categorías generales: simétricos, controlador-switch y asincrónicos.

Tabla 1.

Tipos de Mensajes en OpenFlow v1.0

Tipo de Mensaje	Categoría	Subcategoría
HELLO	Simétrico	Inmutable
ECHO_REQUEST	Simétrico	Inmutable
ECHO_REPLY	Simétrico	Inmutable
VENDOR	Simétrico	Inmutable
FEATURES_REQUEST	Controlador-Switch	Configuración de Switch
FEATURES_REPLY	Controlador-Switch	Configuración de Switch
GET_CONFIG_REQUEST	Controlador-Switch	Configuración de Switch
GET_CONFIG_REPLY	Controlador-Switch	Configuración de Switch
SET_CONFIG	Controlador-Switch	Configuración de Switch
PACKET_IN	Asincrónico	NA
FLOW_REMOVED	Asincrónico	NA
PORT_STATUS	Asincrónico	NA
ERROR	Asincrónico	NA
PACKET_OUT	Controlador-Switch	Cmd desde el controlador
FLOW_MOD	Controlador-Switch	Cmd desde el controlador
PORT_MOD	Controlador-Switch	Cmd desde el controlador
STATS_REQUEST	Controlador-Switch	Estadísticas
STATS_REPLY	Controlador-Switch	Estadísticas
BARRIER_REQUEST	Controlador-Switch	Barrera
BARRIER_REPLY	Controlador-Switch	Barrera
QUEUE_GET_CONFIG_REQUEST	Controlador-Switch	Configuración de Colas
QUEUE_GET_CONFIG_REPLY	Controlador-Switch	Configuración de Colas

Fuente: (Goransson, Black, & Davy, 2014).

2.3.1.2. OpenFlow v1.1

OpenFlow versión 1.1 fue lanzado el 28 de febrero de 2011. Esta versión de la especificación incorpora una serie de características nuevas como múltiples tablas de flujo y soporte para grupos de tablas. Desde el punto de vista práctico la versión 1.1 de OpenFlow tuvo poco impacto y sirvió únicamente para acelerar el proceso de publicación de la versión 1.2. Esto sucedió debido a que su lanzamiento fue justo antes de que se creara la ONF y la comunidad de SDN esperó la primera

versión después de la transición antes de crear implementaciones. Sin embargo, es importante tomar en cuenta esta versión ya que las posteriores se basan en algunas de las características principales de OpenFlow v1.1.

La versión 1.1 aumenta de forma significativa la sofisticación del procesamiento de paquetes en OpenFlow. El cambio más destacado se debe a la adición de múltiples tablas de flujo. El concepto de una tabla de flujo única no difiere de la versión 1.0; sin embargo, en la versión 1.1 es posible retrasar el procesamiento adicional de paquetes a la coincidencia posterior en otras tablas de flujo. Por esta razón es necesario romper la ejecución de las acciones desde la asociación directa con una entrada de flujo. (OpenFlow, 2011)

Con la versión 1.1, el nuevo objeto de protocolo de instrucción está asociado a una entrada de flujo. La línea de procesamiento de 1.1 ofrece una flexibilidad mucho mayor que la disponible en la versión 1.0. Esta mejora se deriva del hecho de que las entradas de flujo se pueden encadenar mediante una instrucción en una entrada de flujo que apunta a otra tabla de flujo. Esto se llama instrucción GOTO. Cuando se ejecuta una instrucción de este tipo, se invoca nuevamente la función de correspondencia de paquetes, esta vez iniciando el proceso de emparejamiento con la primera entrada de flujo de la nueva tabla. Este nuevo canal se refleja en la función en la función ampliada de correspondencia de paquetes que se muestra en la Figura 15.

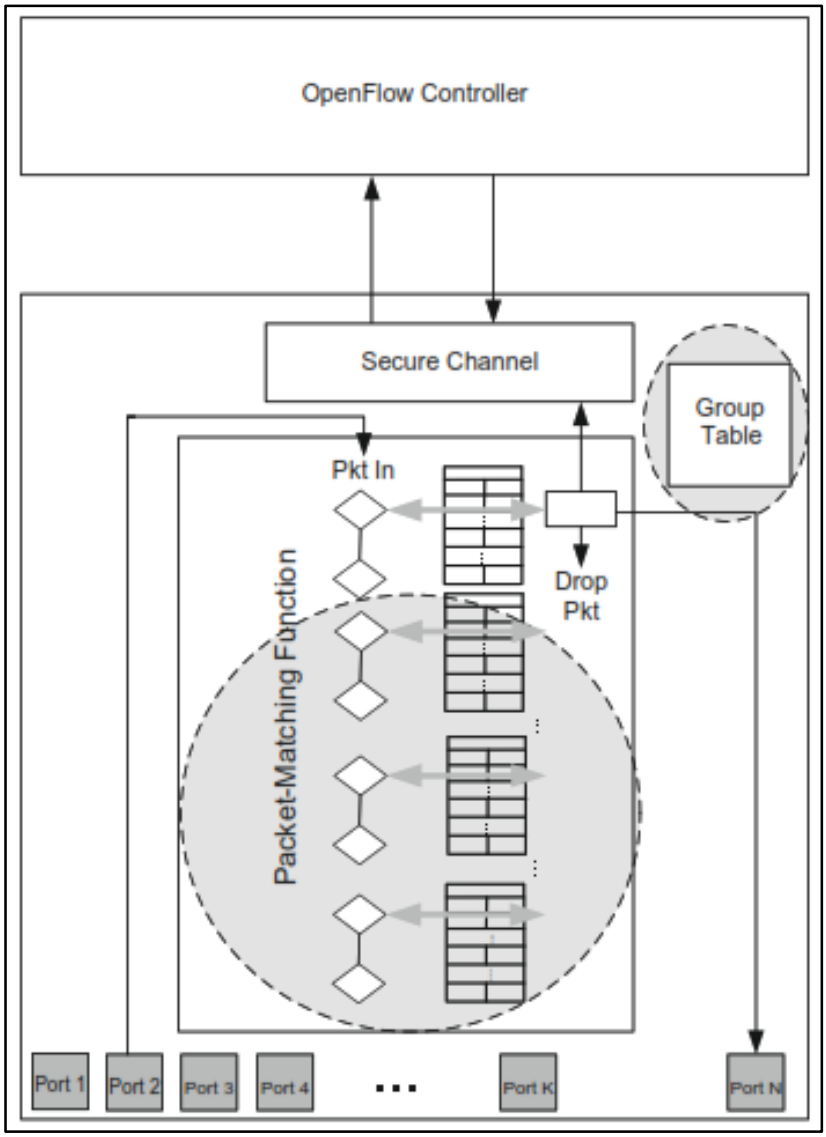


Figura 15. Switch OpenFlow 1.1 con procesamiento de paquetes expandido
Fuente: (Goransson, Black, & Davy, 2014).

El set de instrucciones de OpenFlow versión 1.1 forman el canal que controla las acciones que se toman y en qué orden. Puede hacerse de dos maneras, primero se pueden agregar acciones a un conjunto de otras. El conjunto de acciones se inicializa y modifica con todas las instrucciones ejecutadas durante un paso determinado a través de la ruta trazada. Las instrucciones eliminan o fusionan nuevas acciones dentro del conjunto establecido. Cuando termina la interconexión, las acciones que estaban dentro del conjunto se ejecutan en el siguiente orden:

- Copie TTL de entrada
- Pop (Des apilar Etiquetas)
- Push (Apilar Etiquetas).
- Copiar TTL de salida.
- Decremento TTL.
- Establecer: aplicar todas las acciones SET_FIELD al paquete.
- QoS: aplicar todas las acciones de QoS al paquete.
- Grupo: si se especifica acción de grupo, aplique a los segmentos relevantes.
- Salida: si no se especifica ninguna acción grupal, reenvíe el paquete al puerto especificado.

La versión 1.1 ofrece la abstracción del grupo como una extensión más rica a la opción FLOOD. En OpenFlow 1.1, está la tabla de grupos, que consta de entradas de grupo cada una con uno o más segmentos de acción. Los depósitos de un grupo tienen acciones asociadas que se aplican antes de que el paquete se reenvíe al puerto definido por ese depósito. En v1.1 se pueden lograr los refinamientos de inundaciones, como el multicast, definiendo los grupos como conjuntos específicos de puertos. Algunas veces se puede usar un grupo cuando solo hay un único puerto de salida, como se ilustra en el caso 2 de la Figura 16.

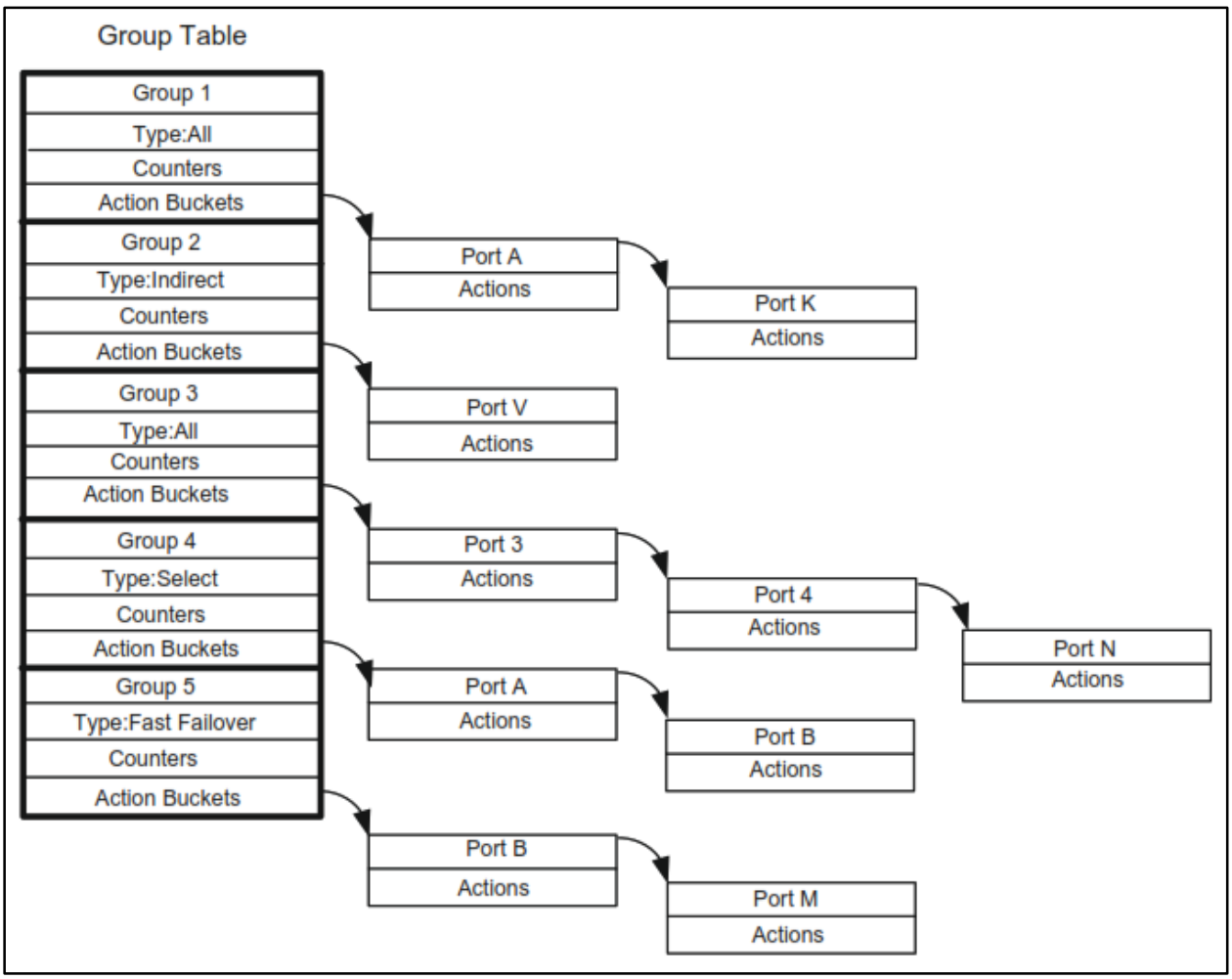


Figura 16. Tabla de Grupo en OpenFlow v1.1
Fuente: (Goransson, Black, & Davy, 2014).

En la versión 1.1 de OpenFlow es la primera que se brinda compatibilidad total con VLAN. Dado que proporcionar soporte completo para múltiples niveles de etiquetas de VLAN requiere un soporte sólido para hacer saltar y empujar múltiples niveles de etiquetas, se integra también el soporte para etiquetado MPLS. Para proporcionar el soporte genérico para VLAN fue necesaria la inserción de nuevas acciones de encadenamiento PUSH y POP. (OpenFlow, 2011)

En esta versión se afirma concretamente que se admite completamente MPLS y el etiquetado VLAN, pero ese soporte requiere la especificación de una lógica de coincidencia muy compleja

que debe implementarse cuando tales etiquetas se encuentran en el paquete entrante. El soporte de coincidencia ampliable proporciona la semántica más generalizada para el controlador, de modo que esta compleja lógica de coincidencia es el switch pueda eliminarse.

En la versión 1.0 el concepto de un puerto de salida se asigna directamente a un puerto físico, con algún uso limitado de puertos virtuales. Aunque la tabla y otros puertos virtuales existían en versiones anteriores de OpenFlow, el concepto de puertos virtuales se aumentó en la versión 1.1. Un conmutador 1.1 clasifica los puertos en las categorías de puertos estándar y puertos virtuales reservados. (Goransson, Black, & Davy, 2014)

Los puertos estándar consisten en:

- **Puertos Físicos**
- **Puertos virtuales definidos por el conmutador:** Se usan para procesamiento más complejo en el paquete que la simple manipulación de la cabecera.

Los puertos virtuales reservados consisten en:

- **ALL:** Este es el mecanismo para inyectar paquetes fuera de los puertos estándar, excepto el puerto por el que llego el paquete.
- **CONTROLLER:** Reenvía el paquete al controlador en un mensaje OpenFlow.
- **TABLE:** Procesa el paquete a través del procesamiento normal. Solo se aplica para paquetes enviados desde el controlador mediante PACKET_OUT.
- **IN_PORT:** Proporciona una función de bucle invertido. El paquete se envía de vuelta al puerto en el que llegó.

- **LOCAL(opcional):** Proporciona un mecanismo mediante el cual el paquete reenvía al software de control local OpenFlow. Puede usarse como puerto de entrada o de salida permitiendo implementar conexiones con el controlador directamente.

- **NORMAL(opcional):** Dirige el paquete por el canal normal no OpenFlow del switch. Solo se puede usar como puerto de salida.

- **FLOOD (opcional):** El uso general de este puerto es enviar el paquete a todos los puertos estándar, excepto al puerto que llegó.

La pérdida de conectividad entre el conmutador y el controlador es una posibilidad seria y real, y la especificación de OpenFlow necesita especificar como se debe manejar. La cache de flujo de emergencia se incluyó en la versión 1.0 para manejar tal situación, pero el soporte para esta caché fue eliminado en la versión 1.1 y se reemplazó con dos mecanismos nuevos, modo seguro fallido y modo independiente fallido. El switch 1.1 ingresa inmediatamente a uno de estos modos al perder comunicación con el controlador. El modo al que se ingrese depende de cual sea compatible con el switch o, si ambos son compatibles, con la configuración de usuario.

En el caso de modo seguro de falla, el switch seguirá funcionando como un switch 1.1 normal, excepto que todos los mensajes destinado al controlador se eliminan. En el modo autónomo de falla, el conmutador también interrumpe el procesamiento de OpenFlow y continúa operando en su modo conmutador o enrutador nativo. Cuando se restablece la conexión, el switch reanuda su modo de operación normal. El controlador al detectar la perdida y restauración puede optar por eliminar las tablas de flujo existentes y configurar el cambio de nuevo. (OpenFlow, 2011)

2.3.1.3. OpenFlow v1.2

OpenFlow 1.2 fue lanzado el 5 de diciembre de 2011, se define como la evolución de las versiones anteriores 1.0 y 1.1. La familia de estándares OpenFlow incluye una serie de nuevos componentes como el protocolo de configuración (OF-Config), especificaciones de prueba e interoperabilidad para proporcionar rendimiento y ajuste según sea necesario.

En OpenFlow 1.2, la especificación del conmutador describe los formatos y protocolos mediante los cuales un conmutador OpenFlow recibe, reacciona y responde a los mensajes de un controlador OpenFlow. La especificación 1.2 se basa significativamente en versiones anteriores en muchos aspectos incluyendo mejoras y soporte a las extensiones ya. (OpenFlow, 2011)

La versión 1.2 sigue soportando las acciones de las versiones anteriores, pero una mejora importante proporciona la capacidad de establecer el valor de cualquier campo en el encabezado del paquete que se puede usar para la coincidencia. Esto se debe al hecho de que la misma codificación mejorada está disponible en 1.2 para la configuración generalizada de campos en el encabezado del paquete. Por ejemplo, el soporte IPv6 se elimina naturalmente por el hecho de que cualquiera de los campos que puede describirse por el codificador también puede establecerse utilizando la acción set_field. La capacidad de emparejar un encabezado IPv6 a un nuevo valor en conjunto proporciona compatibilidad para esta característica principal de la versión 1.2. El nuevo cálculo de CRC se realiza automáticamente cuando una acción cambia los contenidos del paquete.

La codificación OXM[1] (Object XML Mapper) se usa para extender un mensaje PACKET_IN enviado desde el switch al controlador. En versiones anteriores, este mensaje incluía los

[1] Función que permite importar/exportar datos de un objeto por medio del mapeo de sus propiedades.

encabezados de paquetes utilizados en la coincidencia que dieron como resultado que el switch decidiera reenviar el paquete al controlador. Además del contenido del paquete, la decisión de coincidencia de paquetes se influencia por la información de contexto. Anteriormente, esto consistía en el identificador del puerto de entrada. En OpenFlow 1.2 esta información de contexto se expande para incluir el puerto virtual de entrada, el puerto físico de entrada y los metadatos que se han creado durante el procesamiento de coincidencia del paquete.

El formato OXM proporciona un medio de transporte conveniente para comunicar el estado de coincidencia de paquetes cuando el conmutador decide reenviar el paquete al controlador. El conmutador puede reenviar un paquete al controlador por las siguientes razones:

- No existe flujo coincidente.
- Se ejecutó una instrucción en curso, ordenando que se envíe un paquete coincidente al controlador.
- El paquete tiene un TTL inválido.

En versiones anteriores de OpenFlow, había un soporte muy limitado para los controladores de respaldo. En el caso que se perdiera la comunicación con el controlador actual, el switch ingresa al modo seguro de falla o al modo independiente de falla. La propuesta de que el conmutador intentara conectar a los controladores de copia de seguridad previamente configurados fue realizada a partir de esta especificación, pero no se describe explícitamente.

En 1.2 el switch puede estar configurado para mantener conexiones simultaneas a múltiples controladores. El switch debe garantizar que solo envíe mensajes a un controlador perteneciente a un comando enviado por ese controlador. En el caso de que un mensaje de cambio pertenezca a

múltiples controladores, se duplica y se envía una copia a cada controlador. Un controlador puede asumir uno de los tres roles relativos a un conmutador y se ilustran en la Figura 17.

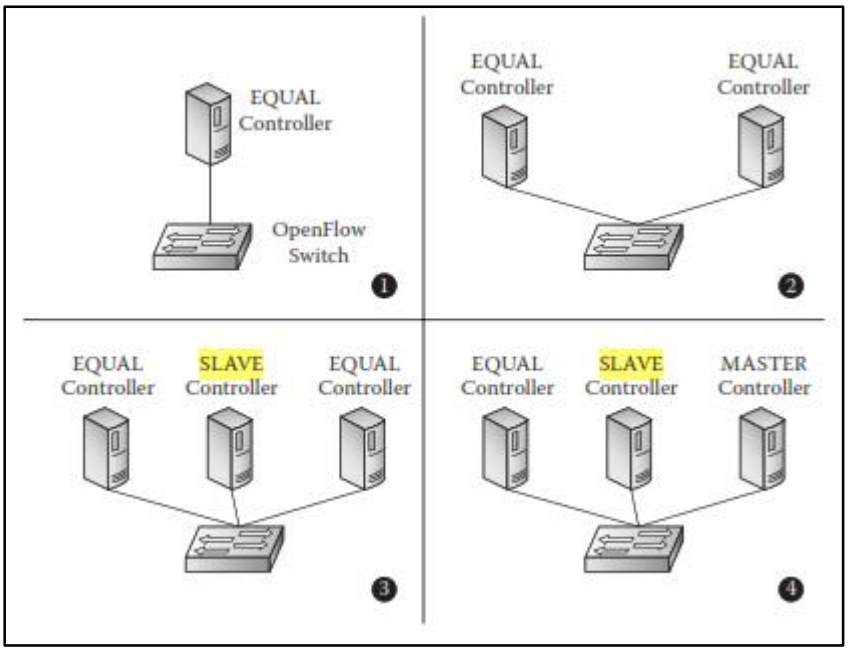


Figura 17. Modos de un Controlador OpenFlow
Fuente: (Morreale & Anderson, 2015).

Los términos maestro, igual y esclavo se relacionan con la media en que el controlador debe cambiar la configuración del switch. El menos potente es evidentemente el modo esclavo, en el que el controlador solo puede solicitar datos al switch, como las estadísticas, pero no se permite que las modifique. Ambos modos el igual y el maestro le permiten al controlador la capacidad completa de programar el switch, pero en el caso del modo maestro, el switch impone que solo un switch este en modo maestro y q todos los demás estén en modo esclavo. La función de múltiples controladores forma parte de la manera en que la especificación OpenFlow aborda el requisito de alta disponibilidad (HA).

2.3.1.4. OpenFlow v1.3

OpenFlow versión 1.3 fue lanzado el 13 de Abril del año 2012. Este lanzamiento fue un hito bastante importante para los desarrolladores, ya que muchas de las implementaciones y despliegues actuales se basaron en esta versión de la especificación para estabilizar y homologar todos los controladores en torno a una sola versión.

Dado que la versión 1.3 representa un gran avance en la funcionalidad y no ha aparecido rápidamente otra versión, brinda la oportunidad para que los diseñadores de ASIC desarrollen soporte para hardware sobre muchas de las características de OpenFlow 1.3 con el fin de lograr un mercado más estable para las nuevas series de equipos.

A pesar de existir una gran oportunidad para los ASIC, algunas características posteriores a la versión 1.0 son difíciles de implementar sobre hardware como el matching iterativo a causa de las tasas de línea. Es probable que los chips de la vida real que soportan la versión 1.3 tengan que limitar el número de tablas de flujo a un número manejable. (Goransson, Black, & Davy, 2014)

En esta versión aparecen un par de mensajes MULTIPART_REQUEST y MULTIPART_REPLY que reemplazan al mensaje READ_STATE que en versiones anteriores usaban STATS_REQUEST y STATS_REPLY para obtener información. En lugar de tener la información de capacidad incrustada como si fuera estadística de tabla, se utiliza el nuevo par de solicitud-respuesta de mensaje y la información se transmite utilizando un formato de valor de longitud tipo estándar (TLV). Algunas de las capacidades que pueden informarse de esta nueva manera incluyen la siguiente tabla, la entrada de flujo de tabla incorrecta y el experimentador. Los formatos de datos para las capacidades son el formato TLV, y esta información de capacidad se ha eliminado de la estructura de las características de la tabla.

OpenFlow 1.3 amplia la capacidad de manejo limitada de las tablas de flujo mediante la inducción de una entrada incorrecta en la tabla de flujos. El controlador programa una entrada de flujo en un switch de la misma manera que programaría una entrada de flujo normal. La entrada en la tabla de flujos es distinta y por definición de prioridad más baja (cero) que garantiza ser la última entrada de flujo que puede coincidir en la tabla.

La ventaja del manejo avanzado de las tablas de flujo es que la semántica completa de la entrada de flujo en 1.3, incluidas las instrucciones y acciones, puede aplicarse al caso de una falla de tabla. Sin embargo, al usar la semántica de entrada de flujo genérica, ahora es posible tratar fallas de tabla de forma más sofisticada, incluyendo pasar el paquete que falla a una tabla de flujo numerada más alta para su posterior procesamiento. Esta capacidad agrega más libertad al lenguaje de coincidencia incorporado por las tablas de flujo, las entradas, las instrucciones y acciones asociadas.

La especificación OpenFlow 1.3 presenta un marco de medición más flexible. Los medidores se definen de forma periódica y residen en una tabla de contadores. La Figura 18 muestra la estructura básica de un medidor identificado por su ID y como se relaciona con un flujo especifico. El marco está diseñado para ser extensible para soportar la definición de medidores complejos en futuras versiones de OpenFlow. Tal soporte futuro puede incluir medidores más sensibles que proporcionaran capacidades DiffServ QoS. (Nadeau & Gray, 2013)

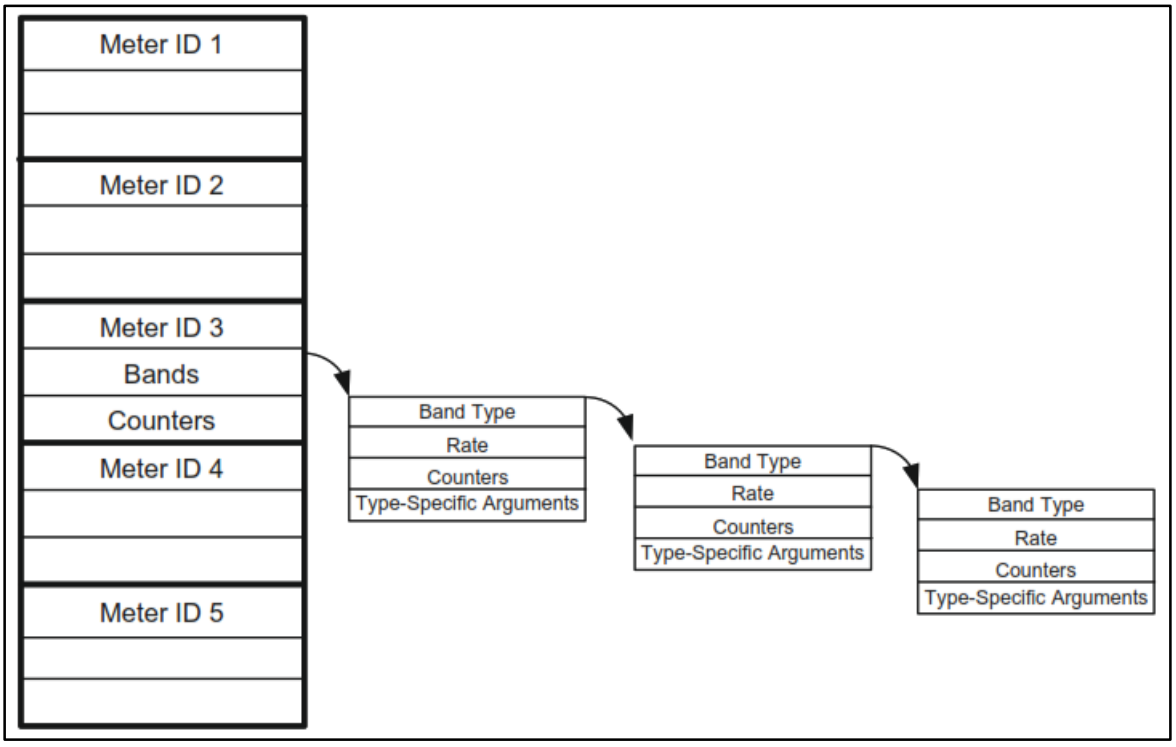


Figura 18. Tabla de comprobación de flujos OpenFlow 1.3
Fuente: (Goransson, Black, & Davy, 2014).

Existen tipos opcionales de medidores descritos en 1.3 que consisten en observaciones DROP y DSCP. La observación de DSCP indica que el campo DiffServ DROP precedente del campo DSCP debe reducirse, lo que aumenta la probabilidad de que este paquete se elimine en caso de congestión de la cola.

Los algoritmos de limitación de velocidad conocidos y simples, como el cubo con fugas, pueden implementarse de manera directa bajo este marco. Esta función proporciona control directo de QoS en el nivel de flujo del controlador OpenFlow mediante una programación cuidadosa de los medidores.

Respecto a las versiones anteriores de la especificación, OpenFlow 1,3 introduce un mensaje SET_ASYNC que permite al controlador especificar los tipos de mensajes asíncronos que está

dispuesto a recibir de un conmutador. Además, permite que el controlador filtre ciertos códigos para no recibir información de un origen. Un controlador puede usar dos filtros diferentes: uno para el rol maestro o igual y otro para el rol esclavo. Esta capacidad del filtro existe conjuntamente con la capacidad de habilitar o deshabilitar mensajes asíncronos de forma periódica. Esta nueva capacidad está orientada al controlador y no orientada a los flujos.

En esta versión se introduce una capa adicional de paralelismo al permitir conexiones múltiples por canal de comunicaciones. Es decir, entre un único controlador y un switch, pueden existir conexiones múltiples. La Figura 19 representa las múltiples conexiones y canales entre un switch y un controlador MASTER.

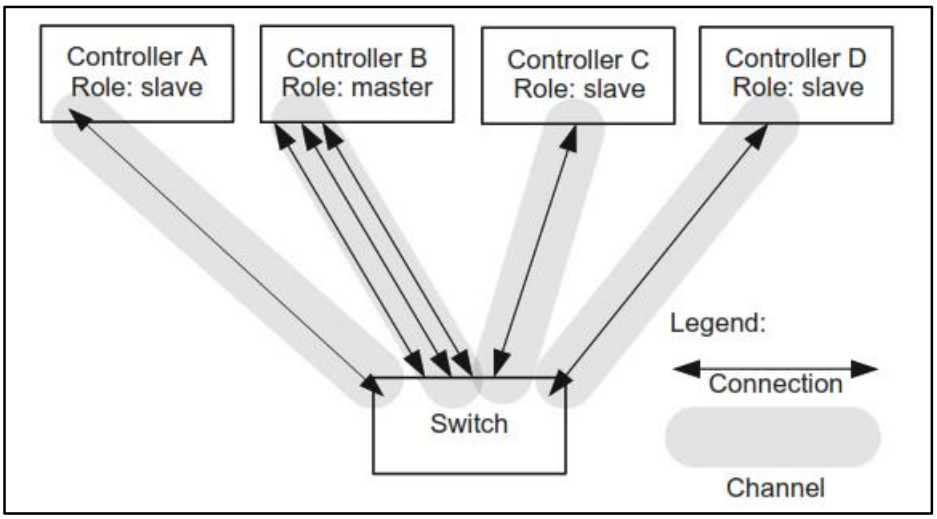


Figura 19. Múltiples conexiones y canales en un switch.
Fuente: (Goransson, Black, & Davy, 2014).

El manejo sencillo de un mensaje PACKET_IN por parte del controlador implica la realización de una función completa de coincidencia de paquetes para determinar el flujo existente al que se relaciona este paquete, si lo hay. A medida que crece el tamaño de las implementaciones comerciales de OpenFlow, las consideraciones de rendimiento se vuelven cada vez más

importantes. En consecuencia, la especificación evolutiva incluye algunas características que están meramente diseñadas para aumentar el rendimiento en situaciones de gran ancho de banda. (Morreale & Anderson, 2015)

Las versiones anteriores de OpenFlow admitían el etiquetado VLAN y MPLS. Otra tecnología WAN/LAN conocida como bridging de backbone de proveedor (PBB) también se basa en etiquetas similares a VLAN y MPLS. PBB permite a las LAN de los usuarios estar en la capa dos enlazadas a través de los dominios del proveedor, lo que perite una separación de dominio completa entre la capa de usuario de dos dominios y el dominio del proveedor a través del uso de la encapsulación MAC-in-MAC.

2.3.2. CONTROLADOR OPENFLOW

Un controlador OpenFlow es un tipo de controlador de redes definidas por software que usa la especificación OpenFlow. Se considera el punto estratégico en una SDN ya que mediante el uso de OpenFlow conecta y configura los dispositivos de red ya sean enrutadores o conmutadores, para determinar la mejor ruta para el tráfico de la aplicación. También existen otras especificaciones y protocolos de SDN que un controlador es capaz de usar, como OpFlex, Yang y NetCOnf.

Los controladores OpenFlow permiten simplificar las tareas de administración de la red, manejando todas las comunicaciones entre aplicaciones y dispositivos para administrar y modificar efectivamente los flujos de la red para satisfacer las necesidades de los usuarios. Cuando el plano de control esta implementado a base de software, en vez de un firmware o sistema operativo de red, los administradores pueden administrar el tráfico de red de forma más dinámica y en un nivel más específico. Un controlador OpenFlow transmite información a los conmutadores y

enrutadores a través de la API sur y las aplicaciones junto a la lógica transaccional a través de la API norte.

Particularmente, los controladores OpenFlow establecen un punto de control centralizado para supervisar una gran variedad de componentes de red habilitados para OpenFlow como se muestra en la Figura 20. El protocolo OpenFlow está diseñado para aumentar la flexibilidad al eliminar los protocolos propietarios de los proveedores y fabricantes de hardware. (SDxCentral, 2017)

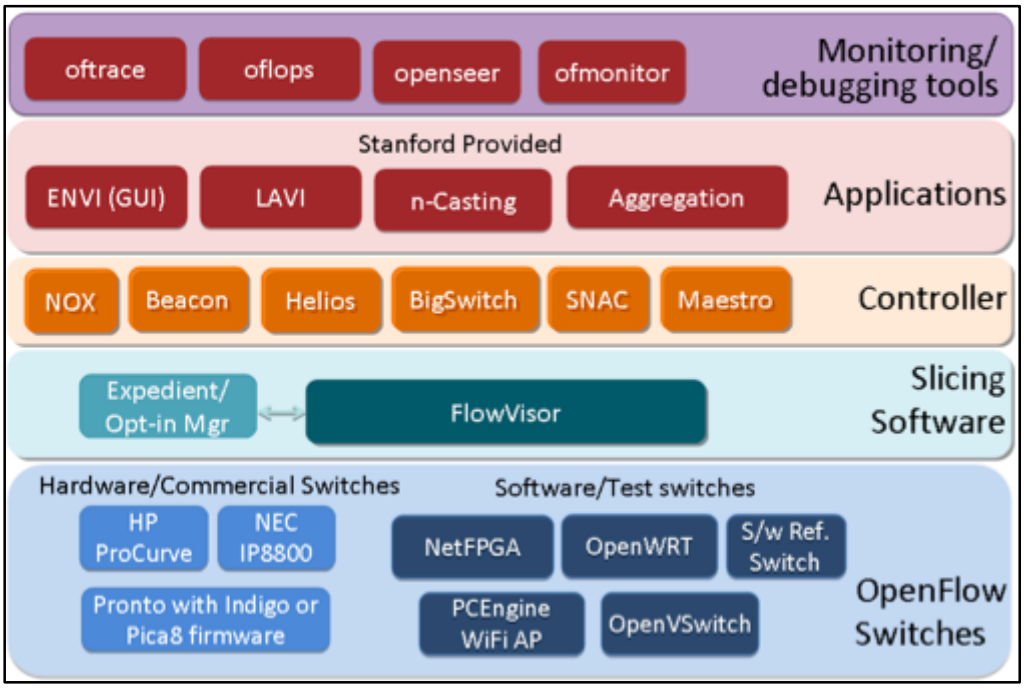


Figura 20. Capacidades de un Controlador OpenFlow.
Fuente: (SDxCentral, 2017).

2.3.3. SWITCH OPENFLOW

Un switch OpenFlow es un programa de software o dispositivo de hardware que es capaz de reenviar paquetes en un entorno de redes definidas por software (SDN). Los switches OpenFlow están basados en la especificación OpenFlow o en su defecto tienen compatibilidad con las funciones que permiten trabajar con flujos.

En un conmutador convencional, el reenvío de paquetes y el enrutamiento de alto nivel ocurren en el mismo dispositivo. En una red definida por software, el plano de datos esta desacoplado del plano de control. El plano de datos todavía se implementa en el switch en sí, pero el plano de control se implementa en el software y un controlador SDN independiente toma decisiones de enrutamiento de alto nivel. El switch y el controlador se comunican a través de los mensajes de la especificación OpenFlow.

La Figura 21 muestra funciones básicas de un conmutador OpenFlow V.1.0 y su relación con el controlador. Como se esperaría en un cambio de paquete, vemos que la función central es tomar paquetes que llegan a un puerto y reenviarlo a través de otro, haciendo cualquier modificación necesaria en el camino. Un aspecto único del conmutador OpenFlow se materializa en la función de correspondencia de paquetes que se muestra en la Figura 21. La tabla adyacente es una tabla de flujo, la flecha doble ancha inicia la lógica de decisión, muestra coincidencia con una entrada en particular en la tabla y dirige el paquete ahora coincidente en un cuadro de acción de la derecha. Este cuadro de acción tiene tres opciones fundamentales para la disposición del paquete entrante:

- A. Reenviar el paquete al puerto local, modificando campos del encabezado
- B. Descartar el paquete
- C. Enviar el paquete al controlador

Las tres rutas fundamentales de paquetes aparecen en la Figura 21. En el caso de la ruta C, el paquete se pasa al controlador por el canal seguro. Si el controlador tiene un mensaje de control o paquete de datos para entregar al switch, el controlador usa el mismo canal seguro, pero en dirección inversa usando el mensaje OpenFlow PACKE_OUT.

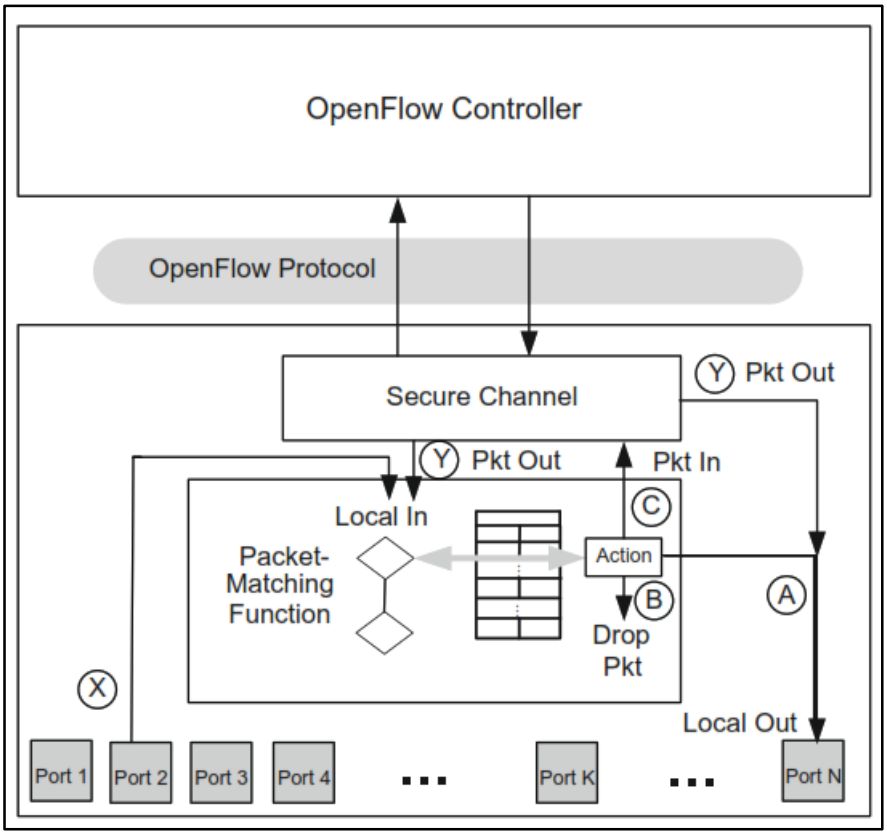


Figura 21. Canales de un Switch OpenFlow V 1.0
Fuente: (Goransson, Black, & Davy, 2014).

2.3.4. TIPOS DE SWITCH OPENFLOW

Los switches compatibles con la especificación OpenFlow vienen en dos tipos: OpenFlow-Only y OpenFlow-Hybrid que también se les llama OpenFlow-Enabled. Los switches OpenFlow- Only solo admiten la operación de esta especificación, en esos switches todos los paquetes son procesados por el canal de OpenFlow, y no pueden procesarse de otra manera.

2.3.4.1. OpenFlow-Only Switch

Los switches OpenFlow-Only son modelos que generalmente están basados en hardware y no son compatibles con las capas 2 y 3 convencionales. Este tipo de switches soporta OpenFlow de manera nativa, contiene múltiples tablas de flujo, y cada flujo contiene múltiples entradas. La

interacción con los paquetes es definida por el procesador y las tablas de flujo instaladas. Requiere que exista al menos una tabla para entrar en funcionamiento, y cuantas menos entradas más simples será el procesado de paquetes en el switch.

2.3.4.2. OpenFlow-Enabled Switch

Los conmutadores OpenFlow híbridos admiten la operación mediante la especificación en sus versiones estables y la operación de conmutación Ethernet, es decir, la conmutación tradicional, el aislamiento de VLAN, el enrutamiento L3, el procesamiento ACL y QoS. Esos switches deben proporcionar un mecanismo de clasificación fuera de OpenFlow que encamine el trafico al canal OpenFlow o al canal normal. Un conmutador OpenFlow hibrido también puede permitir que un paquete vaya desde el canal OpenFlow al canal normal a través de los puertos NORMAL y FLOOD. (The Open Networking Foundation, 2013)

2.3.5. MENSAJES DEL PROTOCOLO OPENFLOW

La comunicación entre el controlador y el switch ocurre usando el protocolo OpenFlow, donde se puede intercambiar un conjunto de mensajes definidos entre estas entidades a través de un canal seguro. El canal seguro es la interfaz que conecta cada OpenFlow switch con el controlador de red. La conexión de Transport Layer Security (TLS) hacia el controlador especificado por el usuario se inicia en el arranque del sistema. El puerto TCP predeterminado del controlador suele ser el 6633. El conmutador y el controlador se autentican mutuamente mediante el intercambio de certificados firmados por una clave privada especifica del sitio. (Azodolmolky, 2013)

Cada conmutador debe poseer la capacidad de ser configurado por el usuario con un certificado para autenticar el controlador y el otro para autenticarse en el controlador (intercambio de

certificado). Una vez que el controlador y el conmutador se han autenticado, OpenFlow define tres tipos de mensajes con sus respectivos subtipos para mantener la comunicación entre las partes:

- Controller-to-switch
- Symmetric
- Asynchronous

2.3.5.1. Controlador-Switch

El controlador inicia los mensajes de controlador a conmutador y los usa para administrar o inspeccionar directamente el estado del conmutador. Este tipo de mensajes puede o no requerir una respuesta del conmutador y se clasifica en los siguientes subtipos.

- **Características:** Tras el establecimiento de la sesión TLS, el controlador envía una solicitud de características al switch. Este responde con un mensaje de características que especifique todas las capacidades admitidas por el conmutador.

- **Configuración:** El controlador puede establecer y consultar parámetros de configuración en el switch. El switch solo responde a una consulta del controlador.

- **Modificación de Estado:** Estos mensajes son enviados por el controlador para administrar el estado de los switches. Se utilizan para agregar, eliminar o modificar las entradas de la tabla de flujo o para establecer las prioridades del puerto del switch. Los mensajes de modificación pueden ser de los subtipos ADD, MODIFY, DELETE o MODIFY and DELETE.

- **Leer Estado:** Estos mensajes recopilan estadísticas de las tablas de fluo de conmutadores, los puertos y las entradas de flujos individuales.

- **Enviar Paquete:** El controlador los utiliza para enviar paquetes desde un puerto específico en el conmutador.

- **Barrera:** Los mensajes de solicitud y respuesta de barrera son utilizados por el controlador para garantizar que se cumplen las dependencias de los mensajes o para recibir notificaciones de las operaciones completadas.

2.3.5.2. Mensajes Simétricos

Los mensajes simétricos son iniciados por los conmutadores o el controlador y enviados sin solicitud. Hay tres subtipos de mensajes simétricos en el protocolo OpenFlow.

- **Hello:** Los mensajes de saludo se intercambian entre el conmutador y el controlador al configurar la conexión.

- **Eco:** Los mensajes de petición y respuesta de eco pueden enviarse desde el conmutador o el controlador, y deben devolver una respuesta de eco. Estos mensajes se pueden usar para indicar la latencia, el ancho de banda y/o la actividad de una conexión entre el conmutador y el controlador.

- **Vendor:** Estos mensajes proporcionan una forma estándar para que los switches OpenFlow ofrezcan funcionalidad adicional dentro del espacio de tipos de mensaje para futuras revisiones de la especificación.

2.3.5.3. Mensajes Asíncronos

Los mensajes asíncronos son iniciados por el switch y se utilizan para actualizar el controlador de eventos de red y los cambios en el estado del switch. Los conmutadores envían mensajes asíncronos al controlador para denotar una llegada de paquete, cambio de estado o un error. Hay cuatro mensajes asincrónicos principales. (Azodolmolky, 2013)

- **Paquete Entrante:** Para todos los paquetes sin entrada de flujo coincidente o si un paquete coincide con una entrada y acción de reenvío al controlador, se envía un mensaje de paquete al controlador. Si el conmutador tiene suficiente memoria para almacenar en búfer los paquetes que se envían al controlador, el mensaje de paquete retiene una fracción del encabezado y una ID de búfer q usara el controlador cuando el switch esté listo para reenviar el paquete. Los conmutadores no compatibles con búfer interno deben enviar el paquete completo al controlador como parte del mensaje.

- **Eliminación de Flujo:** Cuando se agrega una entrada de flujo al conmutador mediante un mensaje de modificación de flujo, un valor de tiempo de espera inactivo indica cuando se debe eliminar la entrada debido a la falta de actividad, y de la misma manera cuando existe un valor de tiempo de espera excedido.

- **Estado de puerto:** Se espera que el switch envíe mensajes de estado de puerto al controlador a medida que cambia el estado de configuración del puerto. Estos eventos incluyen cambios en el estado del puerto según lo especificado por 802.1D[2].

[2] Estándar IEEE para puentes MAC, bridging, el protocolo Spanning Tree y el funcionamiento de redes 802.11 entre otros.

- **Error:** El switch puede notificar al controlador en caso de existir problemas usando mensajes de error.

2.3.6. TABLAS DE FLUJO

En el centro del canal de procesamiento de OpenFlow se encuentran las tablas de flujo. Estas tablas se usan para determinar qué acción, en caso de haber coincidencia, debe tomarse en función de la recepción de determinados paquetes. Las tablas de flujo son una parte importante del canal de procesamiento de paquetes de un switch OpenFlow.

Cada tabla de flujo consta de una serie de entradas de flujo, y cada entrada consta de seis elementos de información. Dos de estos elementos son los campos coincidentes y la prioridad. Los campos de coincidencia son los valores que se comparan con los campos específicos en un paquete recibido para determinar si hay una coincidencia.

Es posible que las entradas de la tabla de flujo múltiple coincidan con el paquete al mismo tiempo. Si esto ocurre, entonces el valor prioridad de la entrada de flujo se usa para determinar que coincidencia se usara para proporcionar las instrucciones que se ejecutaran sobre el paquete. La Figura 22 muestra los pasos que sigue un switch OpenFlow una vez que se ha recibido un paquete.

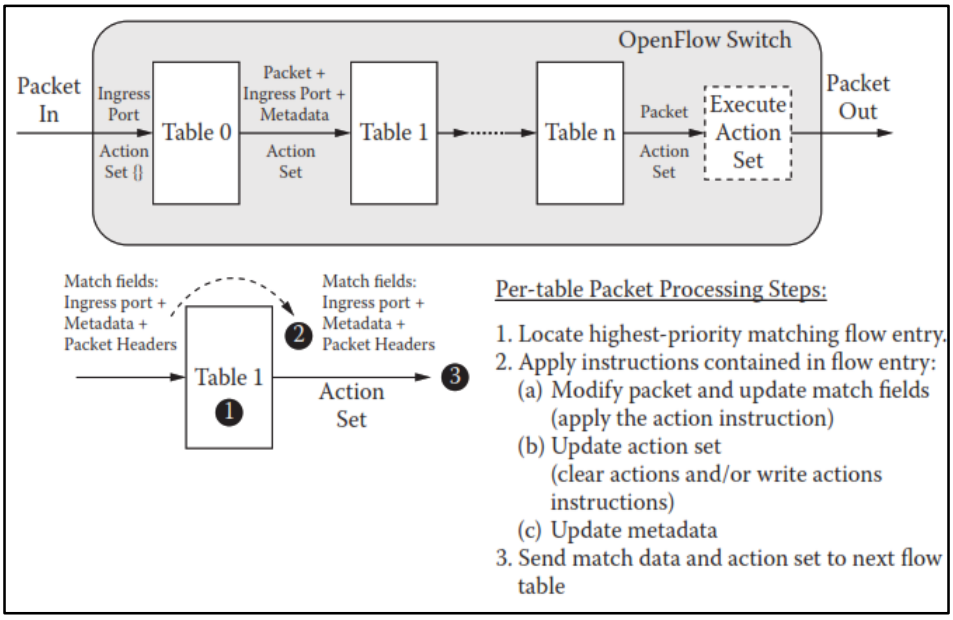


Figura 22. Flujo de paquetes a través del canal de procesamiento OpenFlow
Fuente: (Morreale & Anderson, 2015).

2.3.7. SWITCHES BASADOS EN SOFTWARE

Generalmente los conmutadores son piezas de hardware encargadas de procesar el reenvío de paquetes a nivel de capa 2 y 3. La creciente demanda de ambientes virtuales ha dispuesto el desarrollo de estos dispositivos, pero basados en software para que así puedan ser utilizados con el software de virtualización diseñado para trabajar en ambientes de producción de espacio reducido o también en cloud. Este tipo de conmutador es capaz de sustituir a un conmutador físico manteniendo las mismas funcionalidades, para brindar los servicios de voz y comunicaciones unificadas básicas y profesionales en ambientes productivos. Entre las ventajas que suponen los conmutadores basados en software están la instalación inmediata, no requiere inversiones altas, soporte flexible y mantenimiento simple. Hasta la fecha se han desarrollado gran variedad de proyectos con el objetivo de crear un conmutador estable y completo, pero entre los proyectos más sobresalientes se encuentran aquellos patrocinados por fabricantes como Nicira, VMware, Cisco y la misma ONF.

2.3.7.1. Open vSwitch

Open vSwitch se considera un conmutador virtual multicapa de calidad de producción bajo licencia de código abierto Apache 2.0. Está diseñado para permitir la automatización masiva de redes a través de la extensión programática, al mismo tiempo que admite interfaces y protocolos de administración estándar como NetFlow, sFlow, IPFIX, RSPAN, CLI, LACP, 802.1ag mostrados en la Figura 23. Además, está diseñado para admitir la distribución en múltiples servidores físicos similares al conmutador virtual distribuido vNetwork de VMware o al Nexus 1000V de Cisco.

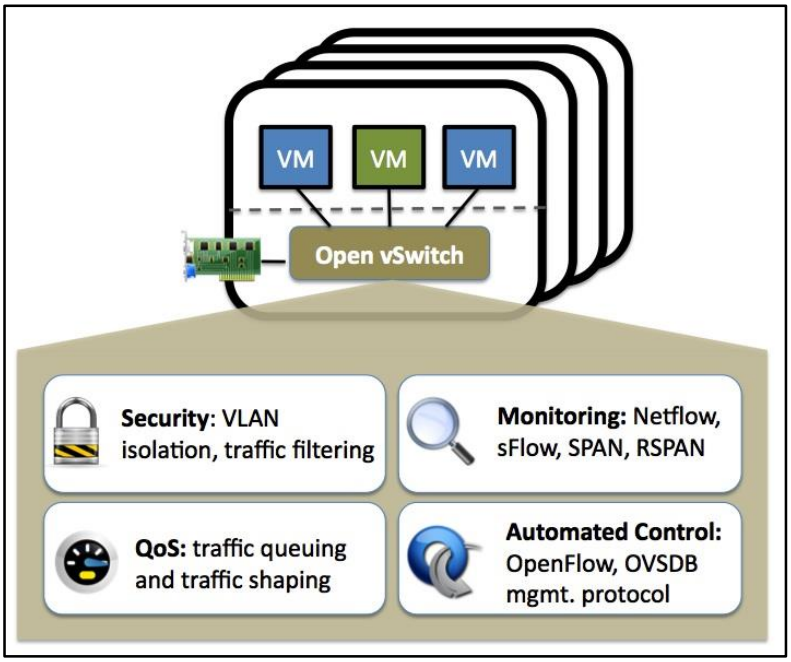


Figura 23. Arquitectura básica de Open vSwitch
Fuente: (Linux Foundation, 2016).

Los hipervisores como Xen, VirtualBox o VMware necesitan la capacidad de puentear el tráfico entre máquinas virtuales y hacia el exterior. Linux Bridge como conmutador L2 incorporado, es un medio rápido y confiable para eso. Pero Open vSwitch está dirigido a implementaciones de virtualización de servidores múltiples, para las cuales Linux Bridge no es una solución adecuada

para la interconexión de las VM's. Los entornos de virtualización de servidores múltiples a menudo se caracterizan por puntos de trabajo altamente dinámicos, mantenimiento de abstracciones lógicas, y la integración de hardware con propósitos especiales.

Todos los estados de la red asociados con una VM deben identificarse fácilmente y migrarse s es necesario entre diferentes hosts físicos. Esto puede incluir un estado suave tradicional, un estado de reenvío L3, ACL, política de QoS o configuración de supervisión, y así sucesivamente. Open vSwitch tiene soporte tanto para configurar, como para migrar estados de red lentos y rápidos entre instancias.

2.3.7.2. Indigo

Indigo es una implementación OpenFlow de código abierto que se ejecuta en conmutadores físicos y utiliza las características de hardware de los circuitos integrados específicos de la aplicación (ASIC) de los conmutadores Ethernet para ejecutar OpenFlow a velocidades de línea. Se basa en la implementación de referencia OpenFlow de Stanford y actualmente implementa todas las características requeridas del estándar OpenFlow 1.0. La implementación de First Generation Indigo Switch ya no es compatible. Indigo2 tiene dos componentes, Indigo2 agente y LoxiGen. El agente Indigo2 representa las bibliotecas principales e incluye una capa de abstracción de hardware (HAL) para facilitar la integración con el reenvío y las interfaces de administración de puertos de conmutadores físicos o virtuales, y una capa de abstracción de configuración para admitir la ejecución de OpenFlow en un modo hibrido en un conmutador físico.

LoxiGen es un compilador que genera bibliotecas OpenFlow en varios lenguajes. Actualmente es compatible con C, pero los lenguajes de programación y scripting de Java y Python se encuentran en desarrollo. El conmutador virtual Indigo (iVS) es un vSwitch liviano y de alto

rendimiento creado desde cero para admitir el protocolo OpenFlow. Está diseñado para habilitar aplicaciones de virtualización de red a gran escala y admite la distribución a través de múltiples servidores físicos usando un controlador habilitado para OpenFlow, similar a VMware vNetwork, Cisco Nexus u Open vSwitch como indica la Figura 24.

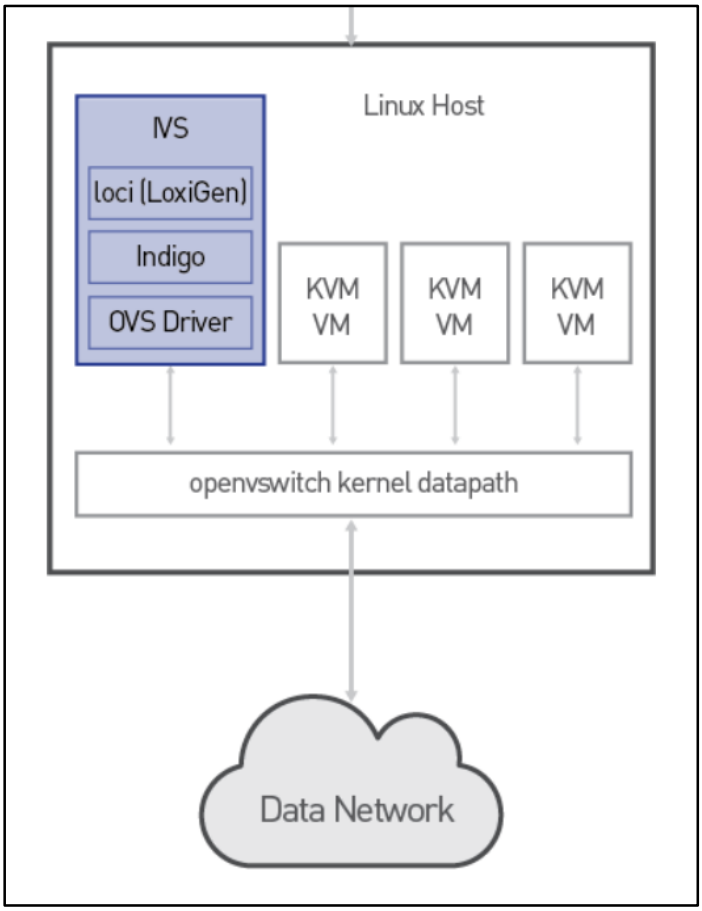


Figura 24. Arquitectura básica de Indigo Virtual Switch
Fuente: (Project Floodlight, 2017).

2.3.7.3. OpenWRT

OpenWRT se describe como una distribución de Linux para dispositivos integrados. En lugar de tratar de crear un único firmware estático, OpenWRT proporciona un sistema totalmente modificable con gestión de paquetes. Esto libera de la selección y configuración de la aplicación

proporcionada por el proveedor y permite personalizar el dispositivo mediante el uso de paquetes para adaptarse a cualquier aplicación como el caso de Pantou que es capaz de habilitar la especificación OpenFlow v.1.3 sobre OpenWRT. Para el desarrollador, OpenWRT es el marco para construir una aplicación sin tener que construir un firmware completo; para los usuarios, esto significa la capacidad de personalización completa, para usar el dispositivo de formas nunca previstas.

Pantou permite convertir un enrutador inalámbrico o punto de acceso comercial en un conmutador habilitado para OpenFlow. OpenFlow se implementa como una aplicación sobre OpenWRT, que es un sistema operativo usado principalmente en dispositivos integrados para enrutar el tráfico de red.

El módulo OpenFlow de OpenWRT se basa en la implementación de referencia de Stanford, convirtiendo el enrutador en un conmutador OpenFlow con funcionalidades dependientes del chipset del dispositivo que comúnmente puede ser Broadcom o Atheros. Es posible construir la imagen con el modulo ya implementado pero se recomienda crear un árbol OpenWRT vanilla antes de agregar funcionalidades relacionadas con OpenFlow. (GitHub Developers, 2013)

2.3.7.4. Of13Softswitch

OF13SoftSwitch es una implementación de conmutadores basados en software en entornos compatibles con OpenFlow 1.3 basada en la implementación de Ericcson TrafficLab 1.1 SoftSwitch con los cambios necesarios en el plano de reenvío para admitir OpenFlow 1.3. El código raíz de este proyecto es la implementación de OpenFlow 1.0 en su versión de la Universidad de Stanford. Los siguientes bloques de construcción están incluidos en el paquete:

- Implementación del switch OpenFlow 1.3: ofdatapath
- Canal seguro para conectar el switch al controlador OpenFlow: ofprotocol
- Biblioteca de software para convertir el canal de comunicación a OpenFlow 1.3: oflib
- Utilidad para línea de comandos para configurar OF13SoftSwitch desde consola: dpctl

Ese proyecto cuenta con el apoyo del dentro de Innovación de Ericcson en Brasil y CPqD lo mantiene en colaboración técnica con Ericcson Research. Las instrucciones para instalar y descargar el modificador de software, junto con los manuales se alojan en la página del proyecto. La naturaleza de este desarrollo permite que sea probado con una versión compatible del controlador NOX, el complemento disector de Wireshark y el paquete de prueba de OpenFlow antes de ser compilado. (Fernandes, 2017)

2.4. CONTROLADORES PARA SDN

La arquitectura o el framework de un controlador de red ideal se detalla en la Figura 25, en donde se puede apreciar algunas de las características que algunos controladores comerciales y de código abierto comparten y otras que son comparables con funciones similares encapsuladas en paquetes o módulos adicionales.

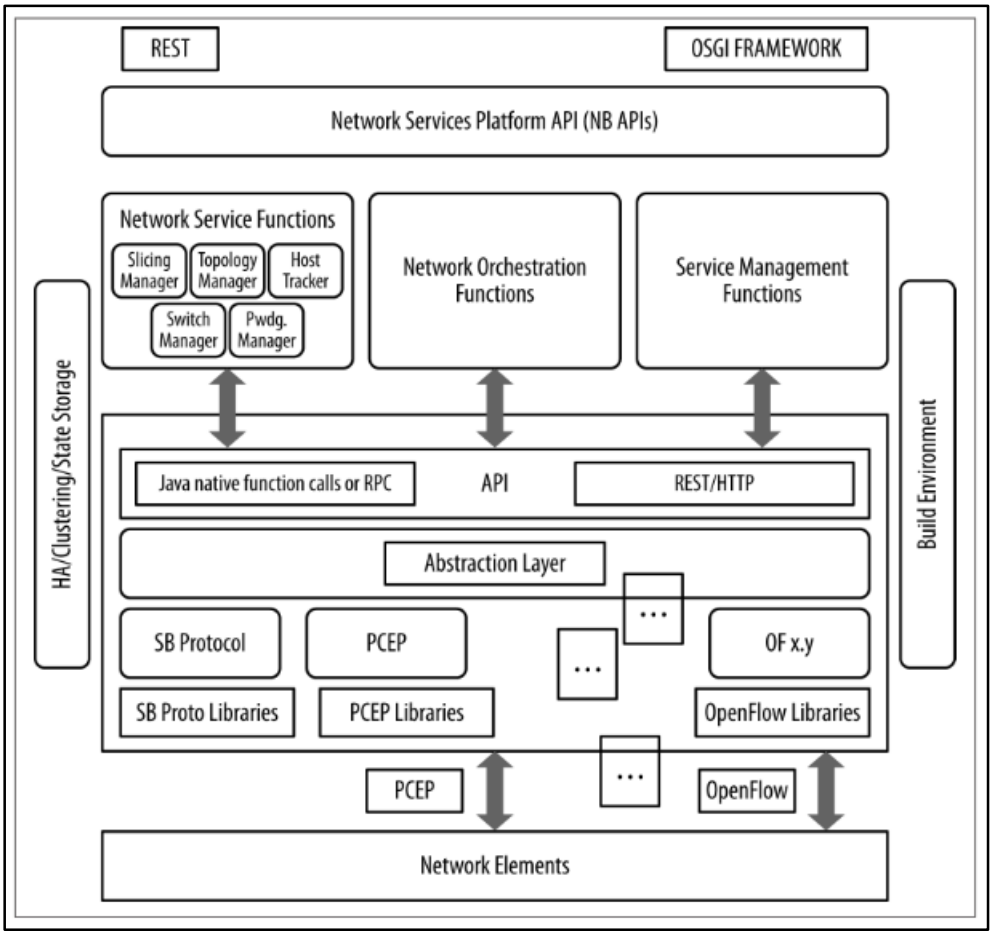


Figura 25. Arquitectura ideal de un framework controlador de SDN.
Fuente: (Nadeau & Gray, 2013).

Una ventaja significativa de SDN es la vista unificada de toda la red, esta es proporcionada exclusivamente por el controlador de red. Sin embargo, el uso de un controlador centralizado rápidamente trae consigo dos problemas de escalabilidad y confiabilidad que se pueden combatir inmediatamente con el despliegue de controladores paralelos para ofrecer redundancia o aún mejor, el control distribuido. (Morreale & Anderson, 2015)

2.4.1. DESCRIPCIÓN GENERAL

La descripción general de un controlador SDN es la de un sistema basado en software o conjunto de sistemas que en contraste pueden proporcionar una serie de servicios o capacidades

que permiten realizar tareas específicas en los dispositivos que son parte de la red definida por software y en algunos casos integrar a los que están fuera del canal de comunicación OpenFlow. Entre las funciones que nos proporciona un controlador están:

- Gestión del estado de la red, y en algunos casos, la gestión y distribución de estado por medio de bases de datos.
- Modelo de datos de alto nivel que captura relaciones entre los recursos administrados, políticas y otros servicios proporcionados por el controlador usando el lenguaje de modelado Yang.
- Interfaz de programación de aplicaciones (API) moderna, a menudo RESTful que publica los servicios del controlador y aplicación.
- Sesión segura de control a través de TCP entre el controlador y los agentes asociados en los elementos de red.
- Protocolo basado en estándares para el aprovisionamiento de estado de red impulsado por aplicaciones en elementos de red.
- Mecanismo de descubrimiento de dispositivos y topología, sistema de cálculo de ruta y potencialmente otros servicios de información centrados en la red o centrados en recursos.

Actualmente, existen diferentes implementaciones de controladores OpenFlow y SDN, la mayoría están construidos a base de recursos de código abierto y otros han sido desarrollados por los grandes fabricantes de hardware como Cisco, Big Switch Networks y VMware. Algunos controladores OpenFlow presentan características que otros no poseen y se hallan escritos en lenguajes diferentes, por lo que la selección de uno de ellos debe hacerse de acuerdo al tipo de implementación que se está buscando. (Nadeau & Gray, 2013)

2.4.2. VMWARE/NICIRA

VMware NSX es la plataforma de seguridad y virtualización de red para los centros de datos definidos por software (SDDC). NSX permite que la red existente escale a un ambiente virtualizado y transforma totalmente todas las transacciones de red. Ofrece un modelo operacional completamente nuevo que permite a los operadores alcanzar mayores velocidades.

De la misma manera que en la virtualización de servidores permite a TI tratar host físicos como un grupo de capacidades de computo, el enfoque de NSX es tratar a la red física como capacidad de transporte que puede consumirse y reutilizarse bajo demanda como indica la Figura 26. La red virtual se convierte en un contenedor de software que presenta componentes lógicos de red en cargas de trabajo conectadas.

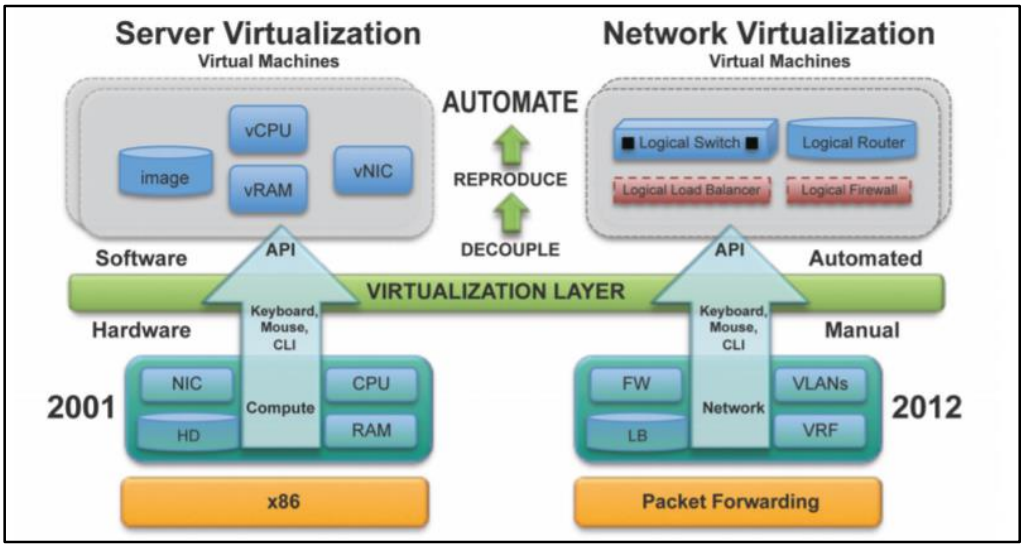


Figura 26. Virtualización de red paralela a la virtualización de servidores.
Fuente: (VMware, 2013).

Las redes lógicas se crean, aprovisionan y administran a base de programación, usando la red física como plano posterior de reenvío de paquetes. Los servicios de red y seguridad se distribuyen y se adjuntan a máquinas virtuales dentro de una red. Cuando una VM se mueve a otro host, estos

servicios son parte de la VM y se mueven con ella. Además, a medida que se agregan nuevas VM's a una red para escalar una aplicación, la política se puede aplicar dinámicamente a las nuevas máquinas virtuales. (VMware, 2013)

2.4.3. CISCO OPEN SDN CONTROLLER

Cisco Open SDN Controller es una distribución comercial de OpenDaylight que brinda agilidad comercial a través de la automatización de la infraestructura de red basada en estándares como indica la Figura 27. Se abstrae de la complejidad de administrar entornos de red heterogéneos para mejorar la prestación de servicios y reducir los costos operacionales. Como software de código abierto, Cisco Open SDN Controller avanza continuamente a través de la innovación continua y el soporte de la comunidad de desarrolladores. (CISCO, 2017)

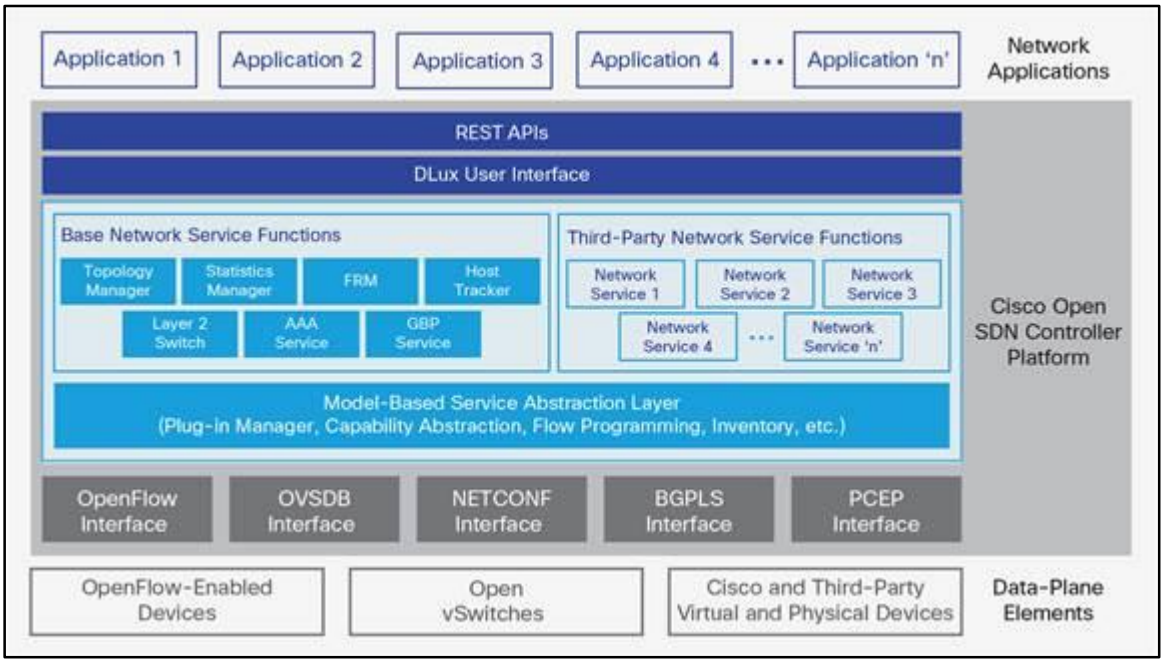


Figura 27. Arquitectura de Cisco Open SDN Controller
Fuente: (CISCO, 2015).

La demanda de aplicaciones está obligando a TI a alcanzar niveles de escala más allá de lo que es posible con las redes tradicionales. Para acelerar la TI, los procesos de aprovisionamiento, configuración, operación y monitoreo de redes deben ser automatizados e instrumentados a través de inteligencia y control basados en la abstracción. Se proporciona soporte robusto de creación, integración y verificación de aplicaciones basadas en el controlador a través del entorno de desarrollo integral de Cisco DevNet[3]. Entre las características y capacidades que presenta Cisco Open SDN Controller están:

- Distribución comercial: distribución reforzada, validada y compatible.
- Clustering: Alta disponibilidad y escalabilidad.
- Facilidad de servicio: supervisión, recopilación de medidas y gestión de registros.
- Empaquetado de Open Virtual Appliance (OVA): instalación simplificada y flexible.
- REST API que admite la integración de aplicaciones a la red.
- Servicios de red API de Java que permiten la creación de funciones incorporadas para entregar capacidades de controlador personalizadas.
- Módulos de dispositivos en dirección sur que conectan elementos de red físicos y virtuales, que admiten entornos de red heterogéneos.

2.4.4. SDN VAN CONTROLLER

El software Controlador de SDN para aplicaciones de red (VAN) de HP proporciona un punto de control unificado en una red habilitada para OpenFlow, lo que simplifica la administración, el aprovisionamiento y la orquestación. Esto permite la entrega de una nueva generación de servicios

[3] Programa de desarrollo de Cisco para ayudar a los profesionales de TI que desean escribir aplicaciones y desarrollar integraciones con productos, plataformas y API de Cisco.

de red basados en aplicaciones. También proporciona interfaces abiertas de programación de aplicaciones (API) para permitir a los desarrolladores de terceros entregar soluciones innovadoras para vincular dinámicamente los requisitos del negocio con la infraestructura de red a través de programas escritos en lenguaje Java personalizados o interfaces de control RESTful de propósito general.

EL controlador VAN SDN está diseñado para funcionar en entornos de campus, centros de datos o proveedores de servicios. Esta plataforma de nivel empresarial entrega una amplia gama de innovadoras funciones de red como compatibilidad con la especificación OpenFlow 1.0 y 1.3, soporte para más de cincuenta modelos de conmutadores HP habilitados para OpenFlow, una API para permitir el desarrollo de terceros, así como una arquitectura extensible, escalable y resistente como describe la Figura 28. (HP Enterprise, 2014)

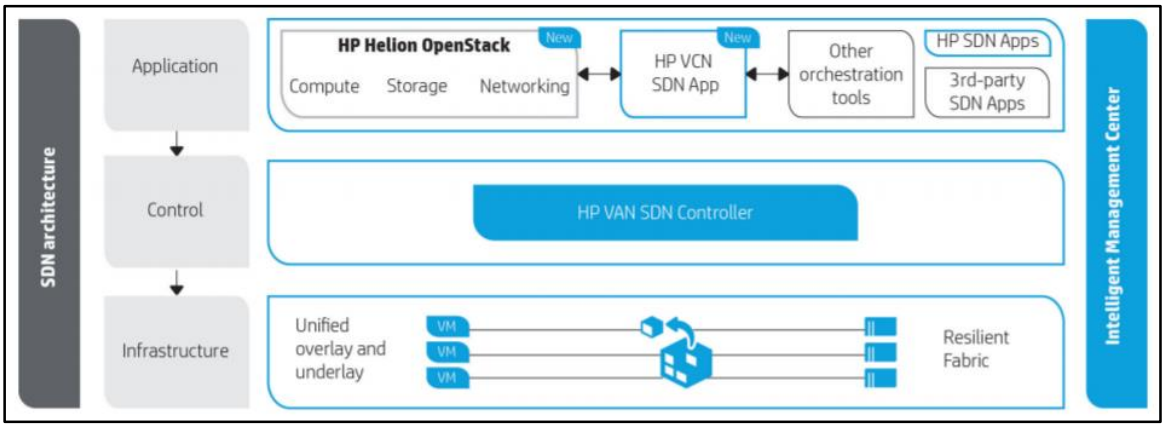


Figura 28. Arquitectura SDN de HP VAN Controller.
Fuente: (HP Enterprise, 2014)**.**

En entornos virtualizados, las herramientas de orquestación en la nube, como OpenStack, ofrecen un mecanismo de alto nivel que facilita el despliegue de aplicaciones. Estas herramientas aprovechan la integración entre infraestructuras, servidores, almacenamiento y redes de nivel inferior. Proporcionar integración con SDN mediante estas herramientas de orquestación, permite

no solo una experiencia de orquestación integrada, sino que unifica las innovaciones físicas y virtuales con SDN.

2.4.5. PLEXXI

Los sistemas Plexxi se basan en el concepto de redes de afinidad, ofreciendo un tipo de controlador ligeramente diferente que cuenta con un algoritmo de optimización de reenvío patentado y un sistema de distribución. La función principal del controlador Plexxi es recopilar información sobre afinidades de forma dinámica desde sistemas externos o estáticamente a través de políticas creadas manualmente y luego traducir esta información de afinidad en topologías de reenvío dentro de la red Plexxi. (PLEXXi, 2017)

La Figura 29 presenta un boceto de la arquitectura de los sistemas Plexxi. La topología física se basa en el anillo y las afinidades se corresponden con los identificadores del anillo, formando así un vínculo estrecho entre los conceptos de superposición y de subcapa. Este vínculo estrecho es más bien un hibrido que tiende a ser una mezcla en una sola capa de red.

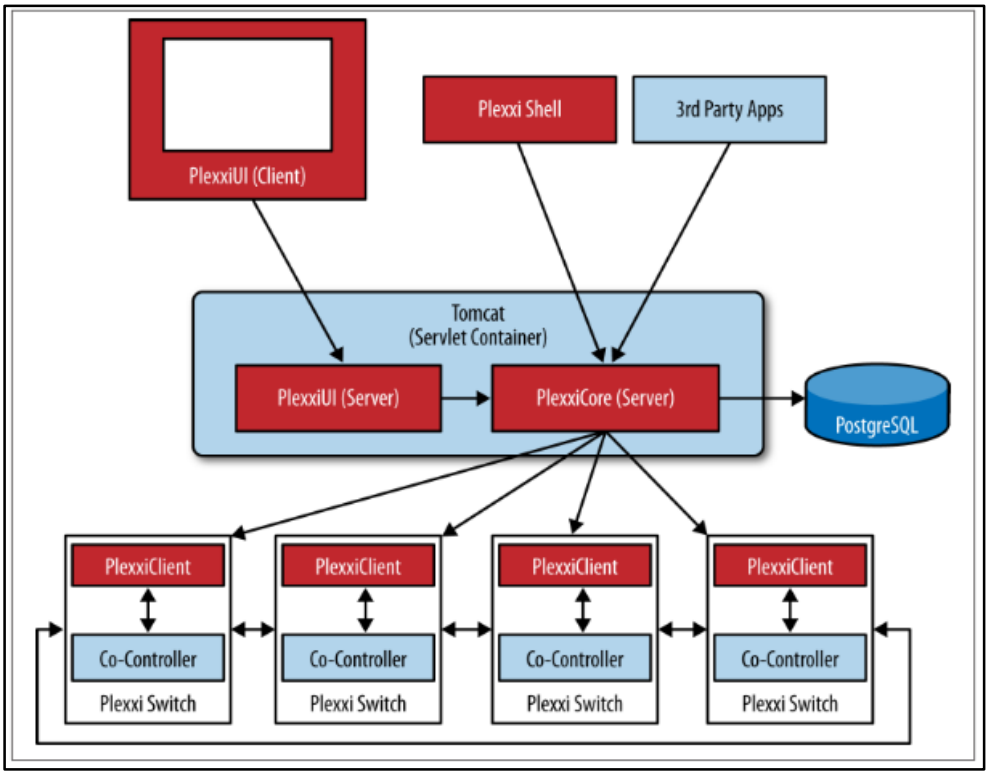


Figura 29. Arquitectura de un sistema Plexxi.
Fuente: (Nadeau & Gray, 2013).

2.4.6. NOX/POX

NOX fue el primer controlador OpenFlow escrito en C++ y proporciona una API para Python. Ha sido la base para muchas investigaciones y proyectos de desarrollo en la exploración temprana del espacio OpenFlow y SDN. NOX posee dos líneas de desarrollo separadas:

- NOX-Classic
- NOX, conocido también como nuevo NOX

La primera es la conocida línea de desarrollo, que contiene soporte para Python y C++ junto con un grupo de aplicaciones de red. Sin embargo, esta línea del desarrollo está en desuso y no hay un plan para un mayor desarrollo en NOX-Classic, pero es mucho más rápido y tiene una base de código mucho más limpia.

POX es una versión de NOX que solo funciona con Python. Se puede considerar como un controlador OpenFlow de fuente abierta escrito en Python, y una plataforma para el desarrollo rápido y creación de prototipos de aplicaciones de red. El objetivo principal de POX es la investigación. Dado que muchos de los proyectos de investigación son de corta duración, el enfoque de los desarrolladores de POX está en las interfaces en lugar de mantener una API estable. NOX y POX se administran en repositorios de código fuente de Git. (GitHub Developers, 2012)

2.4.7. RYU

Ryu que en japonés significa “flujo” es un framework basado en componentes de redes definidas por software. Proporciona recursos de software anclados a una API descrita en la Figura 30, que facilita a los desarrolladores la creación de nuevas aplicaciones de administración y control de red. Es compatible con varios protocolos para administración de dispositivos de red, como OpenFlow, NetConf y OF-config. Ryu admite totalmente la especificación en las versiones 1.0, 1.2, 1.3, 1.4, 1.5 y Nicira. El código se distribuye gratuitamente bajo licencia Apache 2.0.

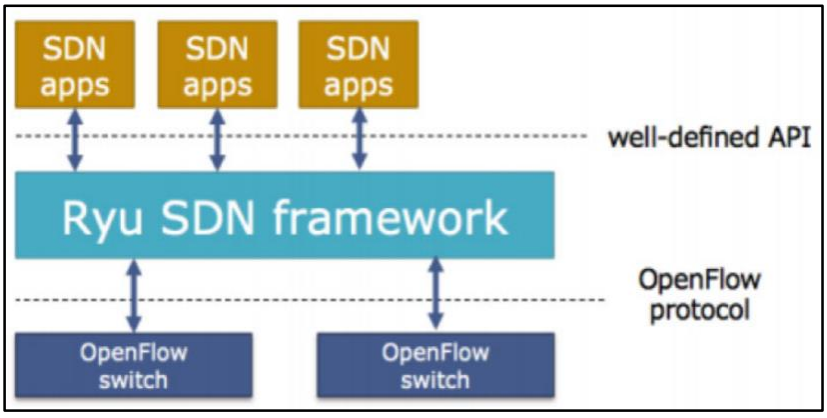


Figura 30. Relación norte y sur del framework RYU.
Fuente: (SDxCentral, 2017).

2.4.8. TREMA

Trema es un marco de programación de controlador OpenFlow capaz de proporcionar los recursos y herramientas necesarias para crear controladores OpenFlow en Ruby. Pone a disposición una biblioteca OpenFlow de alto nivel y también un emulador de red que puede crear redes basadas en OpenFlow para probar en su PC. Este entorno autónomo ayuda a optimizar todo el proceso de desarrollo y prueba. (Trema, 2017)

2.4.9. BEACON

Beacon es un controlador OpenFlow multiplataforma, modular y basado en Java que admite tanto la operación basada en eventos como la operación por hilos. Entre sus características principales se encuentran:

- Estabilidad: Beacon ha estado en desarrollo desde principios de 2010 y se ha usado en varios proyectos de investigación, clases de redes y despliegues de prueba. Actualmente Beacon alimenta un centro de datos experimental con 100 conmutadores virtuales y 20 conmutadores físicos. Durante todo este tiempo se ha mantenido en funcionamiento sin interrupciones.

- Multiplataforma: Beacon está escrito en Java y se ejecuta en muchas plataformas, desde servidores Linux de múltiples núcleos hasta teléfonos Android.

- Código Abierto: Beacon tiene licencia bajo la combinación de la licencia GPL v2 y la excepción de licencia FOSS de la universidad de Stanford v1.0.

- Dinámico: Los paquetes de códigos en Beacon se pueden iniciar, detener, actualizar e instalar en tiempo de ejecución sin interrumpir otros paquetes no dependientes.

- Rápido Desarrollo: Beacon es fácil de poner en funcionamiento. Java y Eclipse simplifican el desarrollo y la depuración de sus aplicaciones.

- Rápido: Beacon posee capacidad multiprocesamiento.

- Interfaz GUI: Beacon incorpora opcionalmente el servidor web empresarial Jetty y un framework GUI extensible y personalizado.

- Frameworks: Beacon se basa en frameworks Java maduros como Spring y Equinox. (Erickson, 2013)

2.4.10. BIG SWITCH NETWORKS/FLOODLIGHT

Floodlight es un controlador OpenFlow de código abierto de clase empresarial, con licencia Apache. Es apoyado por la comunidad de desarrolladores que incluye varios ingenieros de Big Switch Networks. Floodlight está escrito en Java y, por lo tanto, se ejecuta dentro de una JVM. Los componentes arquitectónicos de Floodlight se detallan en la Figura 31.

El controlador Floodlight ha sido probado y admitido por la comunidad de desarrolladores más grande del mundo para controladores SDN. Está diseñado para ser de fácil configuración con dependencias mínimas. Ofrece n sistema modular para desarrolladores, que hace que sea sencillo de ampliar y mejorar.

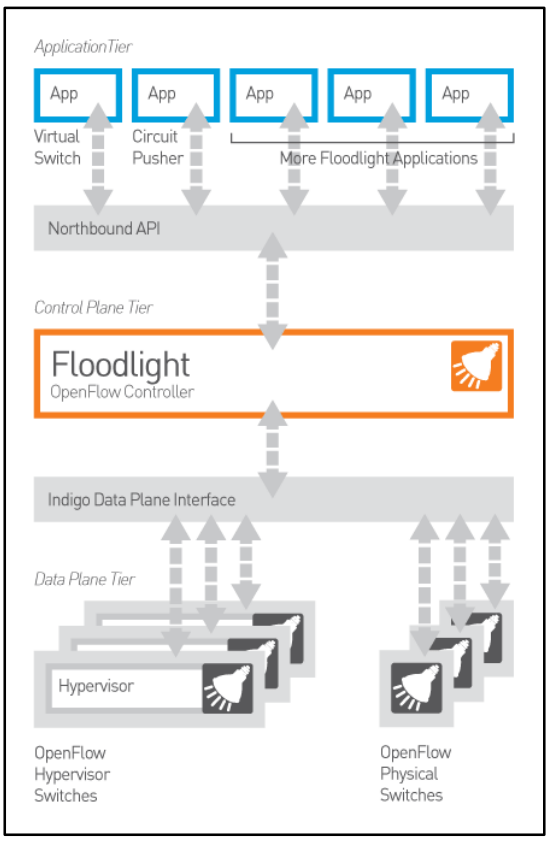


Figura 31. Arquitectura del controlador SDN Floodlight.
Fuente: (Floodlight Project, 2017).

Debido a que OpenFlow es un estándar abierto administrado por la Open Networking Foundation, Floodlight está diseñado para funcionar con el creciente número de conmutadores, enrutadores, conmutadores virtuales y puntos de acceso compatibles con la especificación en sus versiones más recientes. Entre las características destacadas del controlador están:

- Sistema de carga de módulos que hace que sea senillo ampliar y mejorar.
- Fácil de configurar con dependencias mínimas.
- Admite una amplia gama de conmutadores OpenFlow virtuales y Físicos.
- Puede manejar redes hibridas OpenFlow u no OpenFlow, gestionando múltiples islas de switches de hardware OpenFlow.

- Diseño de alto rendimiento con multiprocesamiento desde cero.
- Soporte para OpenStack a través de la plataforma de orquestación de la nube. (Nadeau & Gray, 2013)

2.4.11. OPENDAYLIGHT

OpenDaylight (ODL) es una plataforma modular abierta escrita en Java, creada para personalizar y automatizar redes de cualquier tamaño y escala. El proyecto OpenDaylight surgió del movimiento de SDN, con un claro enfoque en la programabilidad de la red. Fue diseñado desde el principio como una base para soluciones comerciales que abordan una variedad de casos de uso en entornos de red ya existentes. Es un software puro y, como JVM, puede ejecutarse en cualquier sistema operativo o metal desnudo siempre que admita Java. La Figura 32 muestra la estructura del controlador OpenDaylight.

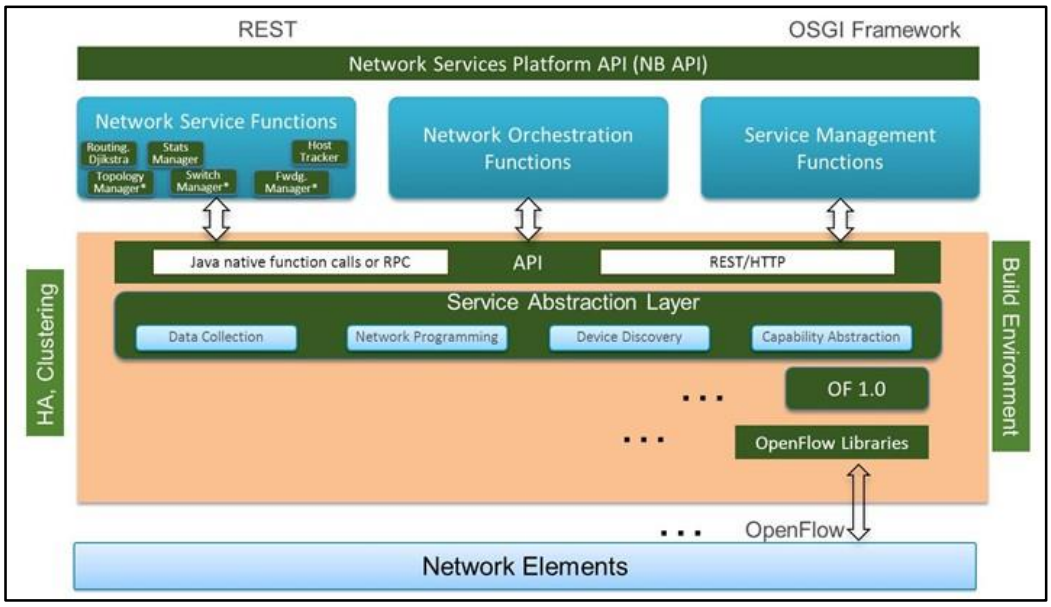


Figura 32. Estructura del Controlador OpenDaylight.
Fuente: (OpenDaylight Project, 2013).

ODL es impulsado por una comunidad global colaborativa de organizaciones de proveedores y usuarios que se adapta continuamente para admitir el conjunto más amplio de casos de SDN y NFV de la industria. Con más de mil desarrolladores, cincuenta organizaciones miembro y soporte para aproximadamente mil millones de suscriptores en el mundo, OpenDaylight está desarrollando rápidamente cadenas de herramientas integradas para casos de uso líderes. El código de OpenDaylight se ha integrado e incorporado en más de cincuenta soluciones y aplicaciones de proveedores, y se puede utilizar dentro de una gama de servicios. También se encuentra en el núcleo de los framework de código abierto más amplios, incluidos ONAP, OpenStack y OPNFV. (OpenDaylight, 2017)

Carbón es la sexta versión estable de OpenDaylight (ODL), proyecto que se ha convertido en la plataforma SDN abierta de facto, brindando asistencia a más de mil millones de suscriptores y empresas líderes. Sus versiones más recientes incorporan soporte en temas clave como mejoras para admitir las necesidades de IoT, gestión integrada de NFV, S3P[4] para permitir la federación y la clusterización como detalla la Figura 33.

[4] Término que engloba cuatro pilares de la automatización de procesos. Sus siglas significan: Scale, Stability, Security and Performance.

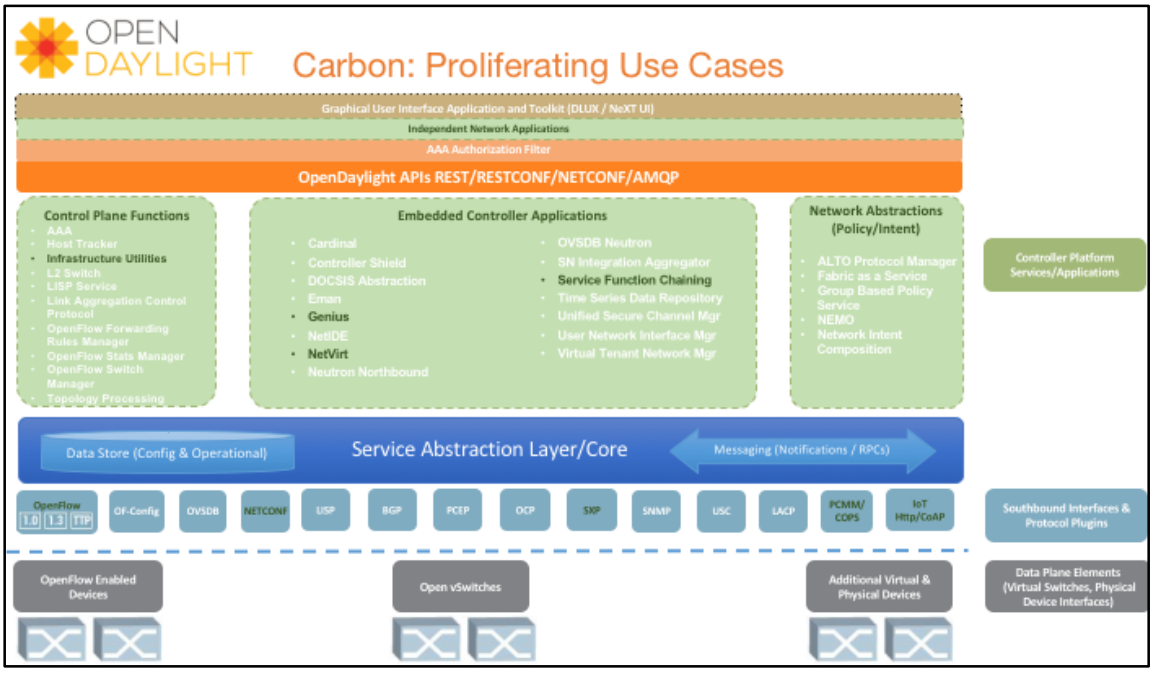


Figura 33. Características y funciones de ODL.
Fuente: (OpenDaylight, 2017).

2.4.12. ONOS

ONOS significa Sistema Operativo de Red Abierta. ONOS proporciona el plano de control para una red definida por software (SDN), administra los componentes de red, como conmutadores y enlaces, y ejecuta programas de software o módulos para proporcionar servicios de comunicación al host finales y las redes vecinas.

ONOS ofrece algunos tipos análogos de funcionalidad, que incluyen API y abstracciones, asignación de recursos y permisos, así como software orientado al usuario, como CLI, GUI y aplicaciones de sistema visibles en la Figura 34. ONOS administra toda la red en lugar de solo algunos dispositivos, lo que puede simplificar enormemente la administración, configuración y despliegue de nuevo software y servicios. La plataforma y las aplicaciones de ONOS actúan como un controlador SDN distribuido, modular y extensible. (Linux Foundation Administrators, 2017)

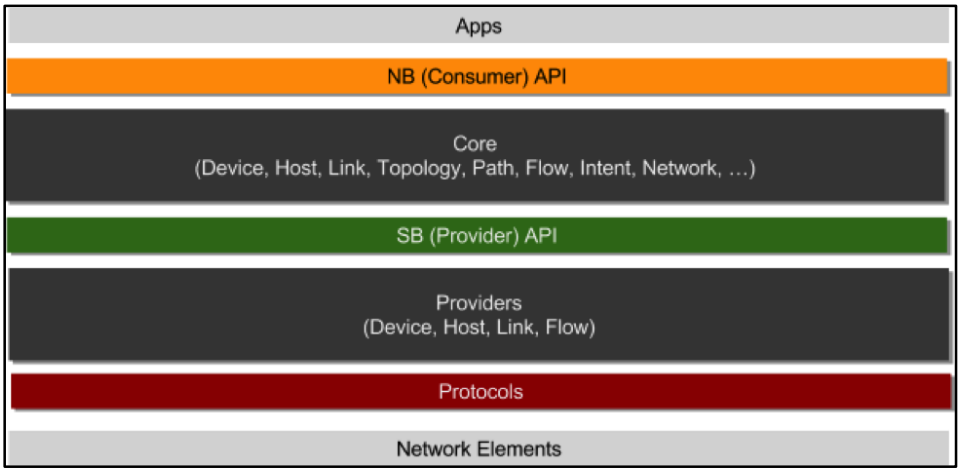


Figura 34. Componentes de ONOS Controller.
Fuente: (ONOS, 2016).

Un servicio es una unidad de funcionalidad que se compone de múltiples elementos que crean una división vertical a través de los niveles como una pila de software. Nos referimos a la colección de mecanismos que integran el servicio como un subsistema. ONOS define claramente varios servicios principales y se muestran organizados en la Figura 35.

- Subsistema de dispositivo: administra el inventario de dispositivos de infraestructura.
- Subsistema de enlace: gestiona el inventario de enlaces de infraestructura.
- Subsistema de host: gestiona el inventario de los hosts y sus ubicaciones en la red.
- Subsistema de topología: administra gráficos de red instantáneos ordenadas por tiempo de vista.
- Servicio de Ruta: calcula y busca rutas entre dispositivos de infraestructura o entre hosts utilizando la gráfica instantánea de la topología más reciente.
- Subsistema para reglas de flujo: administra el inventario de las reglas instaladas de coincidencia o acción para flujos en los dispositivos de infraestructura y proporciona métricas de flujo.

- Subsistema de paquetes: permite a las aplicaciones escuchar paquetes de datos recibidos de dispositivos de red y emitir paquetes de datos a la red a través de uno o más dispositivos de red.

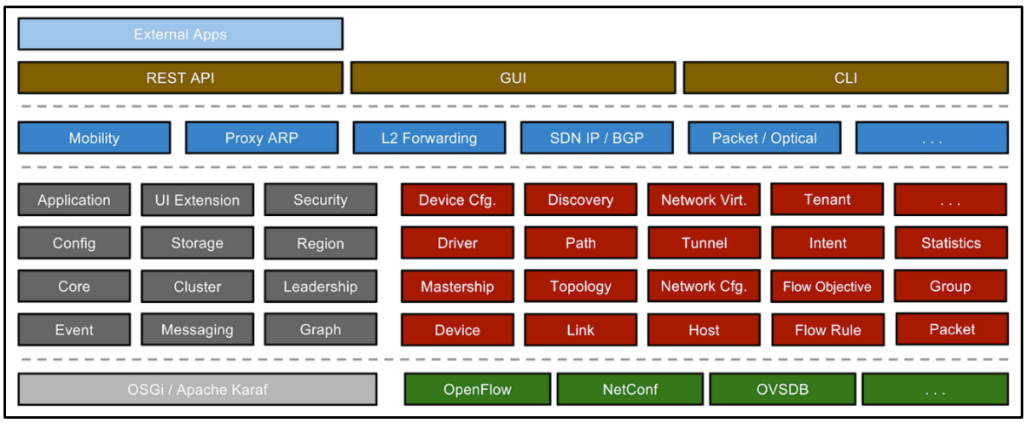


Figura 35. Subsistemas y servicios de ONOS.
Fuente: (Linux Foundation Administrators, 2017).

2.5. CLOUD COMPUTING NETWORKS

La disminución de la importancia del transporte y la aparición de una plataforma de red totalmente IP con demanda de diversos servicios crea oportunidades, o en algunos casos una exigencia para que los proveedores de telecomunicaciones sigan siendo relevantes al adoptar un nuevo framework para la prestación de servicios. Los elementos esenciales del nuevo marco de trabajo incluyen elementos de hardware para la reingeniería en las funciones de software que se ejecutan en la infraestructura de computación en la nube.

La alineación de las redes de acceso tanto en núcleo como en borde usando los patrones de trafico creados por los servicios basados en IP; la integración de las tecnologías de red y nube en una plataforma de software que permita una implementación, administración de servicios rápida con alta automatización; un control definido por software para que tanto la infraestructura como las funciones puedan optimizarse a través del cambio en la demanda del servicio y la disponibilidad

de infraestructura; un aumento en las competencias de integración de software y un modelo de operaciones DevOps[5]. A esto se le conoce como "Network Cloud". (Donovan & Prabhu, 2017)

2.5.1. MODELO SPI PARA CLOUD

La computación en la nube ha recorrido un largo camino desde sus inicios. Los conceptos, el uso y la opinión sobre computación en la nube han cambiado mucho a lo largo de los años. Aunque pocas compañías han comenzado a adoptar con los brazos abiertos, otras todavía están preocupadas por el aspecto de la seguridad.

El modelo SPI se denomina Cloud Computing Stacks en términos técnicos y la explicación más sencilla se puede dar en tres palabras, construir, comprar y desplegar relacionadas en la Figura 36 con cada tipo de Stack. Las siglas SPI son el acrónimo de los modelos más comunes de servicios de computación en la nube, Software como Servicio, Plataforma como Servicio e Infraestructura como Servicio. Cada uno de estos modelos están orientados a diferentes tipos de usuarios y los servicios que se pueden ofrecer a partir de cada uno de ellos varían entre sí. Los servicios ofertados dependen de la demanda para ser desplegados en ambientes de producción.

Figura 36. Modelos de servicio de computación.
Fuente: (IMPRESSICO BUSINESS SOLUTIONS, 2017).

[55] El termino DevOps se asocia a estrategias de transformación digital, y a metodologías como "Continuous Delivery" o desarrollo ágil.

2.5.2. FRAMEWORK PARA SERVICIOS DEFINIDOS POR SOFTWARE

El SDSF es una arquitectura virtual centrada en funciones de red. Utiliza un enfoque estratificado donde las capas se definen en términos de un modelo de servicio en la nube. Se basa en los principios de diseño de la descomposición de funciones de red y la estandarización de la tecnología utilizada para los servicios en tiempo real. Los paradigmas basados en la nube se utilizan para lograr ambas características. Cada capa del SDSF está expuesta a desarrolladores de primera y tercera parte para el diseño y la creación de servicios, por lo que permite nuevas oportunidades comerciales. (Donovan & Prabhu, 2017)

La descomposición de las funciones de red en VNF es una de las tareas posibles sobre el SDSF y permite disponer de tiempo para comercializar los servicios e impulsar el uso eficiente de los recursos de la red y la nube. Permite crear instancias y personalizar solo funciones esenciales según sea necesario para el servicio, lo que hace que la prestación del servicio sea más ágil. Proporciona flexibilidad de dimensionamiento, escalabilidad y también proporciona flexibilidad con el empaquetado e implementación de VNF según sea necesario para el servicio. Permite agrupar funciones en un centro de datos común o en el mismo host físico para minimizar la latencia entre componentes. Además, se pueden seleccionar los mejores proveedores de su clase para la VNF descompuesta. La Figura 37 muestra los principios de SDSF al ilustrar una vista de alto nivel de la descomposición de funciones y mostrando las diferentes capas necesarias para soportar servicios virtualizados.

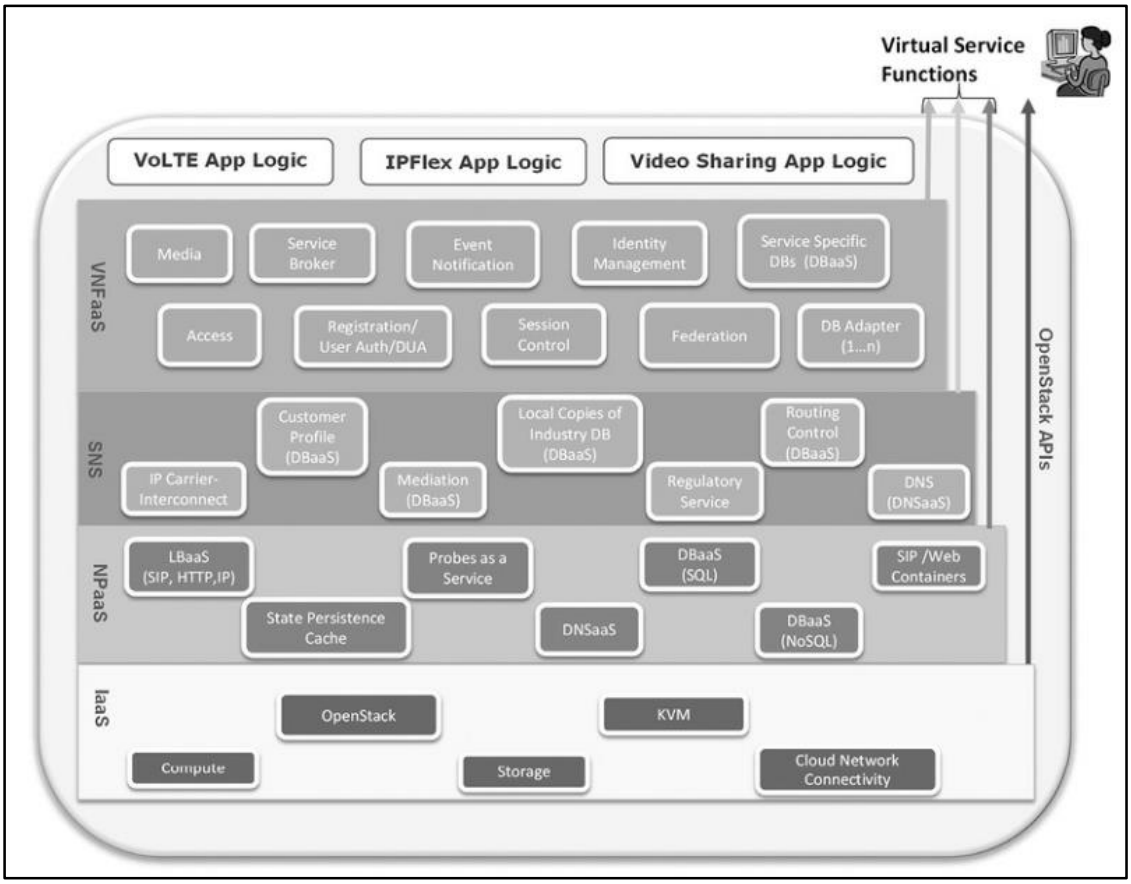


Figura 37. Framework para Servicios Definidos por Software.
Fuente: (Donovan & Prabhu, 2017).

2.5.3. VNF's como Servicio (VNFaaS)

Esta capa incluye VNF descompuestas que se instancian y personalizan de acuerdo al servicio. Forman la base de las VNF descompuestas que forman parte del catálogo disponible para los diseñadores de servicios. En un nivel alto, pueden incluir funciones específicas que la hacen totalmente práctica.

- Funciones de acceso que incluyen funciones de borde de sesión para comunicaciones SIP, web y otras.

- Funciones de medios incluyendo transcodificación de audio, video, reproducción y grabación.
- Control de sesión.
- Registro y autenticación de usuario.
- Funciones de notificación de eventos que permiten que la lógica de la aplicación sea vigilada.
- Gestión de identidad.
- Federación de servicios.
- Agente de servicios.

2.5.3.1. Plataforma de Red como Servicio (NPaaS)

La capa de Plataforma de Red como Servicio expone un conjunto de servicios comunes en la nube que pueden ser utilizados por los servicios en tiempo real. Permite el uso de un conjunto común de tecnologías que se pueden instanciar y administrar de una manera común. Estos servicios pueden estar expuestos a terceros y pueden proporcionar una nueva fuente de ingresos para el operador de telecomunicaciones. Algunos de los servicios de NPaaS que se pueden proporcionar se describen a continuación:

- Base de datos como servicio (DBaaS): los servidores de bases de datos brindan un conjunto de recursos de bases de datos de alto rendimiento, replicación de datos y cache de datos distribuidos. (Hamburguer, 2016)

- Balanceador de carga como servicio (LBaaS): los servicios de equilibrador de carga o intermediario de recursos admiten la capacidad de distribuir solicitudes entre máquinas virtuales en un clúster de VNF y en múltiples clústeres de VNF. (Subramanian & Voruganti, 2016)

- DNS como servicio (DNSaaS): el servicio DNS admite la resolución de direcciones y realiza la traducción de nombres de dominio a direcciones IP numéricas tanto del cloud como del exterior.

- Servicio de persistencia de estado: proporciona persistencia en la nube para los componentes con estado. Es compatible con la tendencia arquitectónica de las aplicaciones sin estado en la nube y permite que las funciones de red se centren en la lógica.

- vTaps como servicio: el servicio vTaps proporciona una forma común de recopilar datos de infraestructura de la nube para el seguimiento de la sesión. Los protocolos que necesitan seguimiento son aquellos que incluyen señalización como SIP, HTTP, RTP y medios de mensajería. (Donovan & Prabhu, 2017)

2.5.4. TODO COMO SERVICIO (XaaS)

La Figura 38 muestra que la relación de los gastos de capital (CAPEX) a los gastos operacionales (OPEX) en equipos de red disminuirá a medida que avanzamos en los primeros años de implementaciones de SDN. Esto se debe a que el crecimiento de SDN coincide con un cambio fundamental en la forma en que los centros de datos y otros grandes consumidores de TI pagaran por sus equipos y servicios de red. Los nuevos modelos de licencia, los modelos basados en suscripción e incluso los modelos basados en el uso son cada vez más comunes.

La rápida expansión de nuevos modelos de prestación de servicios tales como Software como Servicio (SaaS), e Infraestructura como Servicio (IaaS) es un testimonio cambio y del nacimiento del modelo (XaaS) en donde X significa todo, es decir Todo como Servicio. El viejo modelo de empresas que compraban cajas de red amortizándolas como un gasto de capital y repitiendo el ciclo con actualizaciones fue un gran modelo de negocio para los fabricantes, pero esto está desapareciendo gradualmente. Los nuevos modelos, donde prácticamente todos los aspectos del negocio de redes están disponibles como un servicio y se tratan como un gasto operativo, probablemente seguirán desplazando los patrones de gasto de la red tradicional. (Goransson, Black, & Davy, 2014)

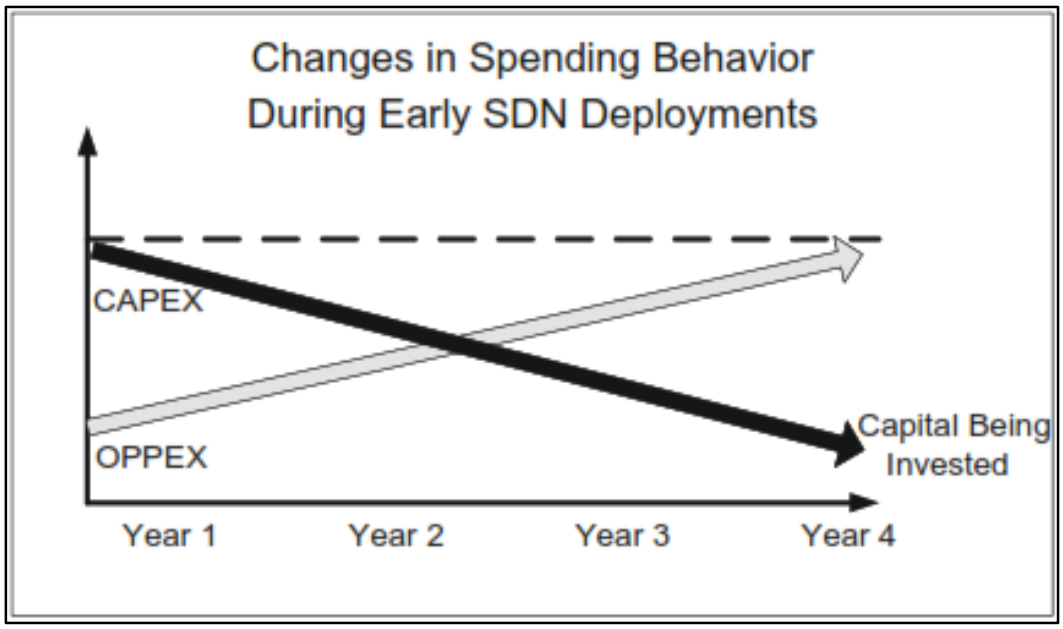


Figura 38. Migración del CAPEX a OPEX en equipos de red.
Fuente: (Goransson, Black, & Davy, 2014).

2.5.5. SDN, NFV y CLOUD

La migración a redes virtualizadas no está exenta de obstáculos. La nueva red se debe acomodar una base existente de tecnologías y dispositivos proporcionados por múltiples proveedores. Los

procedimientos operativos deben revisarse para explotar los beneficios de automatización. La gestión y planificación de la red también se ven afectadas, ya que los enfoques tradicionales se desarrollaron y refinaron asumiendo infraestructura y servicios estáticos. Otra adición importante es el código abierto, cuyo uso en redes va muy por detrás de los dominios de cómputo y almacenamiento. (Subramanian & Voruganti, 2016)

Las redes en nube deben permitir la implementación dinámica y la administración eficiente de máquinas virtuales que admiten cargas de trabajo variadas y cambiantes. Las empresas que buscan crear nubes privadas y nubes hibridas buscan alternativas como OpenStack, la plataforma líder de administración de nube de código abierto para los proveedores de virtualización ya establecidos.

Las redes empresariales existentes suelen ser bastante complejas, generalmente basadas en muchas tecnologías, proveedores y productos. Si bien puede ser tentador construir una pila de redes separada para una nube privada empresarial, la gestión integral de la infraestructura en la nube, otras redes de centros de datos, el campus y la WAN se complementan con equipos, herramientas y métricas de DevOps comunes. Por esta razón, muchas empresas están evaluando una plataforma de arquitectura SDN común que puede abarcar estos dominios y proporcionar visibilidad y control de extremo a extremo.

Los carriers y los Proveedores de Servicios de Comunicación (CSP’s) también se están embarcando en su propio viaje para adoptar la tecnología de virtualización. Los principales operadores están buscando NFV para reducir su dependencia del hardware especialmente diseñado y sus herramientas de administración asociadas, a la vez que mejoran la agilidad del servicio y la capacidad de respuesta operativa a través de la automatización y la inteligencia.

Se necesitan capacidades de red adicionales para virtualizar la entrega del servicio y las operaciones, incluidos gráficos de reenvío de las funciones de red virtualizadas comúnmente denominados encadenamiento de funciones de servicio o SFC, interconexiones de redes incluidos en los límites de los proveedores de servicio y la capacidad de incorporar, desplegar y administrar diversos VNF de muchos proveedores.

2.5.5.1. Orquestación Moderna

Existen muchos recursos de computación, almacenamiento y redes disponibles en los centros de datos. Cuando la virtualización se aplica a estos recursos, la cantidad de recursos asignables explota a gran escala. Asignar estos recursos a los inquilinos del centro de datos de una manera eficiente y manejable es una tecnología compleja en sí misma. Esta tecnología se denomina orquestación.

La orquestación de red está directamente relacionada con la virtualización de la red. El término orquestación comenzó a utilizarse cuando se hizo evidente que la virtualización de la parte de red de un centro de datos complejo, que ya ejecuta virtualmente sus componentes de cómputo y almacenamiento, implicó la coordinación precisa de muchas partes independientes. (Nadeau & Gray, 2013)

A diferencia de las redes heredadas, donde las partes independientes funcionan de manera verdaderamente distribuida y autónoma, la virtualización de red requería que una entidad centralizada coordinara sus actividades en una granularidad muy fina, como un director coordina el momento exacto del interludio en una orquesta sinfónica. Las soluciones de orquestación SDN de código abierto y virtualización de red se resumen en las Tablas 2 y 3.

Tabla 2.

Soluciones de Orquestación de código abierto. Descripción.

Nombre	Descripción
FlowVisor	Crea porciones de recursos de red, delega el control de cada sector, es decir, permite que varios controladores OpenFlow compartan un conjunto de conmutadores físicos.
Maestro	Proporciona interfaces para aplicaciones de control de red para acceder y modificar una red.
OESS	Proporciona el aprovisionamiento de VLAN controlado por el usuario mediante los conmutadores OpenFlow
NetL2API	Proporciona una API genérica para controlar los conmutadores de la capa dos a través de los CLI de los proveedores, no de OpenFlow; utilizable para la virtualización de red no OpenFlow.
Neutron	El componente de red del sistema operativo OpenStack que admite múltiples complementos de red, incluido OpenFlow.

Fuente: (Goransson, Black, & Davy, 2014).

Tabla 3.

Soluciones de Orquestación de código abierto. Detalles.

Nombre	Fuente	Licencia	Lenguaje	Usuarios Probables
FlowVisor	ON.LAB	-	Java	Investigación
Maestro	Rice University	GPL menor	Java	Investigación
OESS	Internet2 Indiana University	Apache 2.0	-	Investigación
NetL2API	Locaweb	Apache	Python	Investigación, Desarrolladores y Operadores
Neutron	Openstack Foundation	Apache	-	Operadores

Fuente: (Goransson, Black, & Davy, 2014)**.**

2.5.5.2. OpenStack

Oenstack es una amplia plataforma de código abierto para la computación en la nube, lanzada bajo la licencia Apache. La Figura 39 muestra el rol de OpenStack, así como sus componentes. OpenStack proporciona la virtualización de los tres componentes principales del centro de datos:

computo, almacenamiento y redes. La función de cálculo se llama Nova, la de almacenamiento está compuesta por Swift y Cinder.

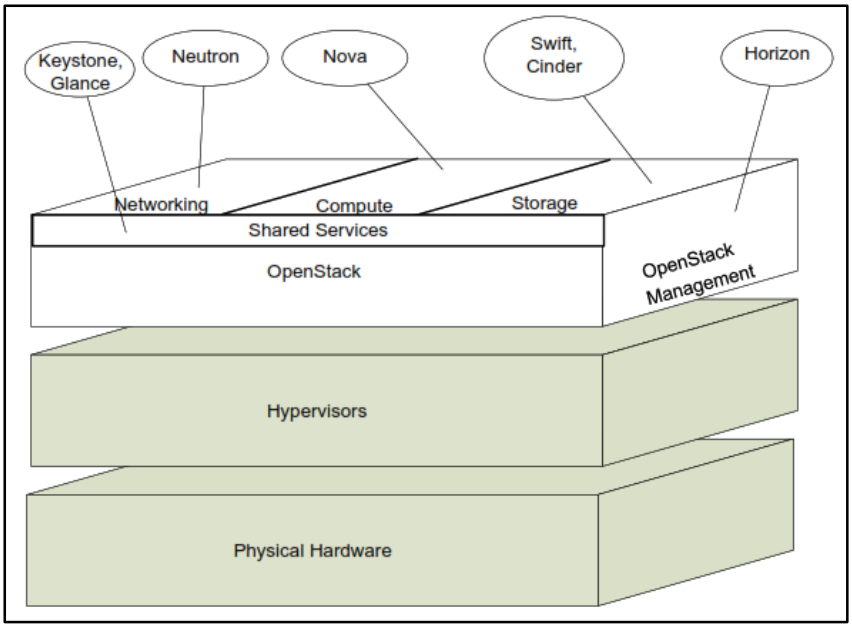


Figura 39. Rol y componentes de OpensStack.
Fuente: (Goransson, Black, & Davy, 2014).

El componente de virtualización de red de OpenStack es proporcionado por Neutron y, por lo tanto, es el componente más relevante para SDN. Neutron anteriormente era conocido como Quantum y arquitectónicamente, el papel de Neutron en OpenStack se materializa como una serie de complementos que proporcionan la interfaz entre la red y el equilibrio de los componentes de computación en la nube de OpenStack.

Aunque OpenStack no está limitado a usar un único complemento de red, se incluye soporte para varias opciones. La Figura 40 se observa que el complemento Neutron pude interactuar con la API hacia el norte de un controlador OpenFlow. De esta forma, Neutron puede proporcionar la capa de abstracción de red para una red habilitada para OpenFlow. (Goransson, Black, & Davy, 2014)

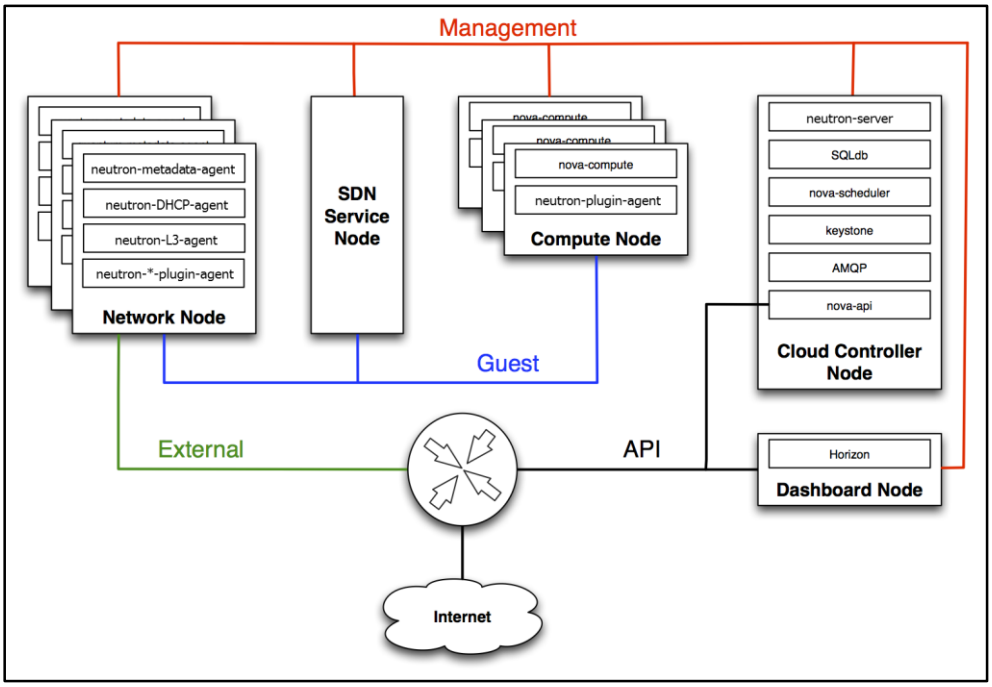


Figura 40. Plug-in y agentes de OpenStack Neutron.
Fuente: (OpenStack, 2017).

Así como OpenFlow puede trabajar con múltiples aplicaciones diferentes de control de red a través de una API hacia norte, también puede Neutron de OpenStack tener muchos tipos diferentes de complementos de red. OpenStack y OpenFlow pueden integrarse para proporcionar una solución de red holística para la computación en la nube, pero ninguno esta exclusivamente vinculado al otro. Como se muestra en la Figura 41, OpenStack puede usar complementos de Neutron para controlar dispositivos de red heredados, un controlador OpenFlow que controlad conmutadores físicos habilitados para OpenFlow o conmutadores virtuales como Open vSwitch.

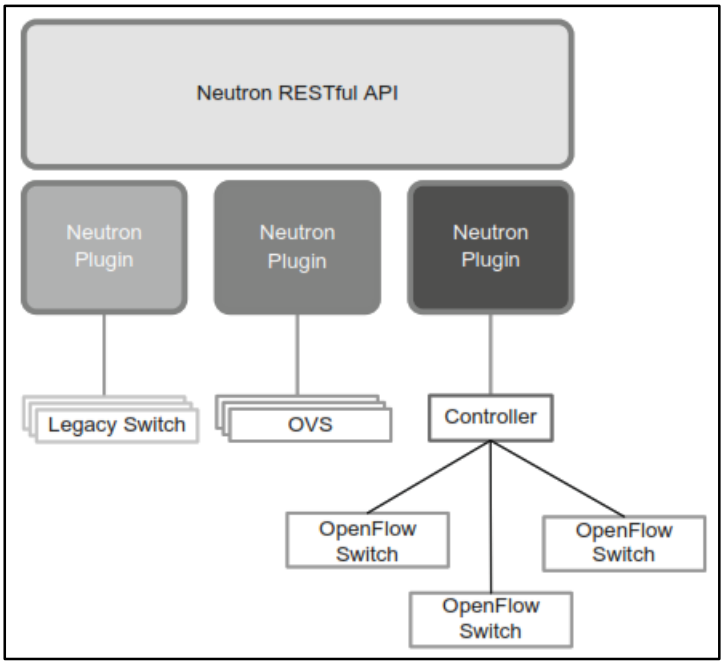


Figura 41. Integración de complementos de Neutron.
Fuente: (Goransson, Black, & Davy, 2014).

2.5.5.3. CloudStack

Apache CloudStack es un software de código abierto diseñado para implementar y administrar grandes redes de máquinas virtuales, como una plataforma de computación en la nube altamente escalable de Infraestructura como servicio (IaaS). CloudStack es utilizado por una serie de proveedores de servicios para ofrecer servicios de nube publica, y por muchas empresas para proporcionar una oferta en la nube privada local, o como parte de una solución de nube hibrida.

CloudStack es una solución integral que incluye toda la pila de funciones que la mayoría de las organizaciones desean con la nube IaaS: orquestación informática, Network-as-a-Service, administración de cuentas y usuarios, una API nativa completa y abierta, contabilidad de recursos y una interfaz de usuario (UI) de primera clase como se muestra en la Figura 42.

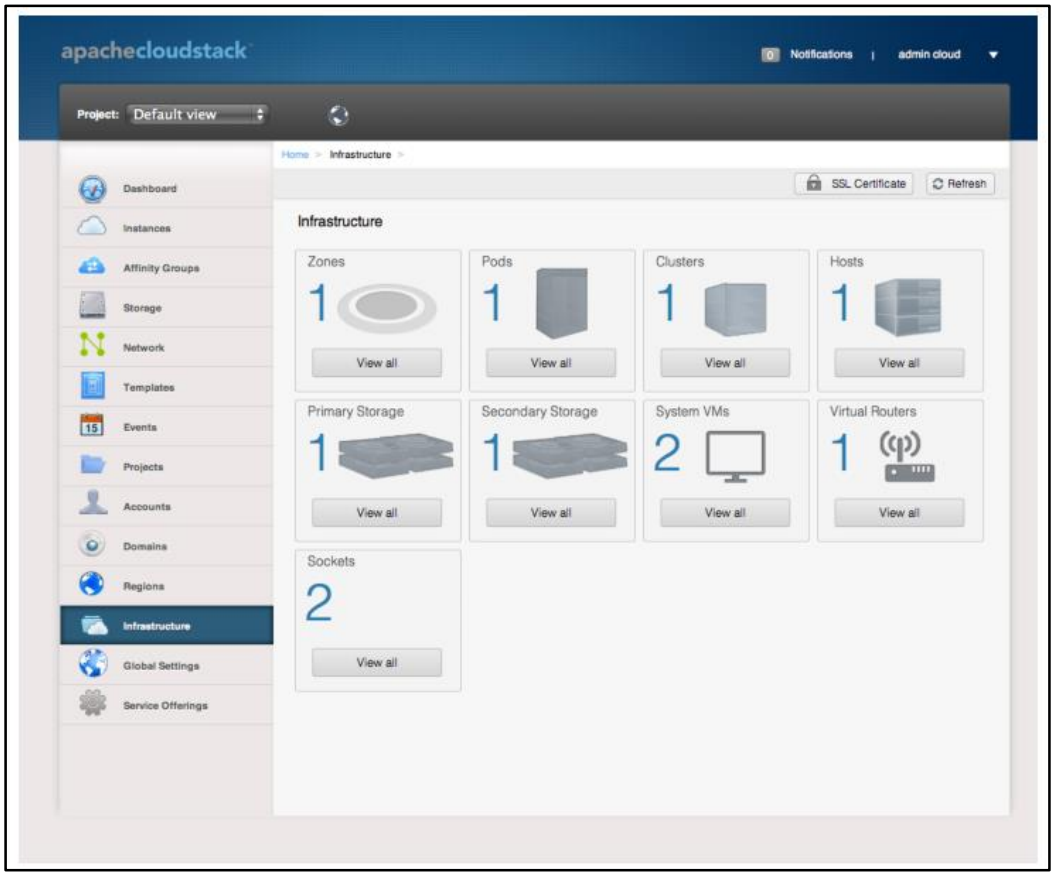


Figura 42. Dashboard de Apache CloudStack.
Fuente: (Goasguen, 2014).

Actualmente CloudStack admite su uso con los hipervisores más populares: VMware ESXi, KVM, Citrix, XenServer, Xen Cloud Platform (XCP), Oracle VM Server y Microsoft Hyper-V. Los usuarios pueden administrar su nube con una interfaz web fácil de usar, herramientas de línea de comando y una API RESTful con todas las características. Además, CloudStack proporciona una API que es compatible con AWS EC2 – EC3[6] para las organizaciones que desean implementar nubes hibridas. (Apache Software Foundation, 2017)

[6] Servicio web que proporciona capacidad informática en la nube segura y de tamaño modificable. Está diseñado para facilitar a los desarrolladores el uso de la informática en la nube a escala de la web.

2.5.5.4. PackStack

PackStack es una utilidad de línea de comandos que utiliza módulos Puppet[7] para admitir la implementación rápida de OpenStack en servidores existentes a través de una conexión SSH. PackStack es adecuado para desplegar instalaciones de prueba de concepto de un solo nodo e instalaciones de múltiples nodos más complejas. Las opciones de implementación se proporcionan de forma interactiva, a través de la línea de comandos o a través de un archivo de texto que contiene respuestas pre configuradas a las preguntas de PackStack. (RedHat Customer Portal, 2017)

2.6. CLUSTERING

Un clúster de servidores es un grupo de dos o más servidores llamados nodos o miembros, que ejecutan un sistema operativo específicamente para procesamiento de información por volúmenes dentro de un centro de datos, y trabajan juntos como un único sistema para proporcionar una alta disponibilidad de servicios para los clientes. Cuando ocurre una falla en uno de los miembros del clúster, los recursos se redirigen y la carga de trabajo se redistribuye a otro miembro en el clúster. (Red Hat, Inc., 2007)

2.6.1. TIPOS DE UN CLÚSTER

Se puede usar clústeres de servidores para garantizar que los usuarios tengan acceso constante a recursos importantes basados en el servidor. Los clústeres de servidores esta diseñados para aplicaciones que tienen un estado en memoria de larga ejecución o datos actualizados frecuentemente. Los usos típicos para los clústeres de servidores incluyen servidores de archivos, servidores de impresión, servidores de bases de datos y servidores de cómputo. Los cuatro tipos

[7] Herramienta de código abierto escrita en Ruby, que permite la automatización de servidores.

principales de clúster son: almacenamiento, alta disponibilidad, balanceo de carga y alto rendimiento. (Microsoft TechNet, 2003)

Los clústeres de almacenamiento proporcionan una imagen consistente del sistema de archivos en los miembros del clúster, lo que permite que los servidores lean y escriban simultáneamente en un solo sistema de archivos compartido. Un clúster de almacenamiento simplifica la administración del almacenamiento al limitar la instalación y el parcheo de las aplicaciones a un sistema de archivos. Además, con un sistema de archivos de todo el clúster, se elimina la necesidad de copias redundantes de los datos de aplicación y simplifica la copia de seguridad y la recuperación de desastres como ilustra la Figura 43.

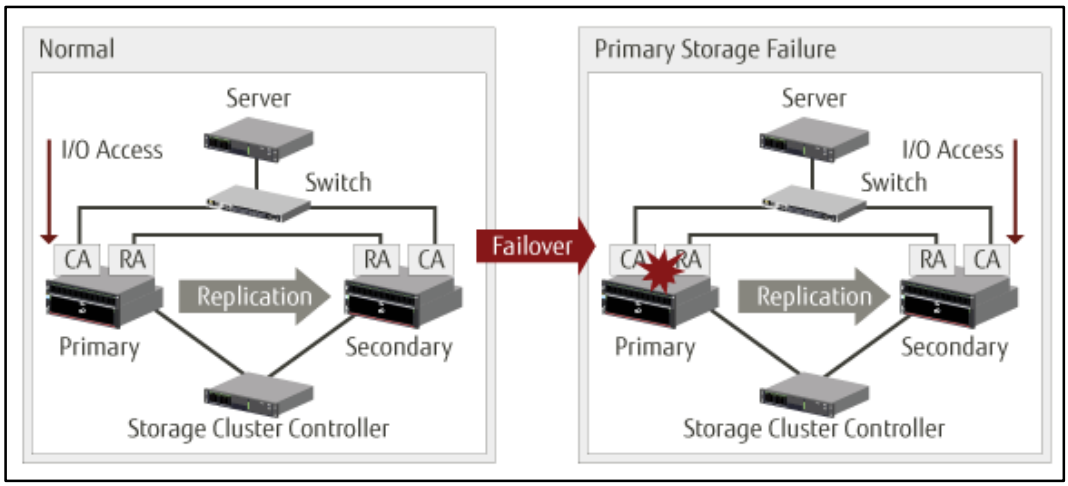


Figura 43. Ventaja de un clúster de almacenamiento ante un sistema normal.
Fuente: (FUJITSU, 2017).

Los clústeres de alta disponibilidad proporcionan disponibilidad continua de servicios al eliminar los puntos únicos de falla y al estropearse los servicios de un nodo del clúster, otro respalda el funcionamiento en caso de que un nodo quede fuera de servicio como se muestra en la Figura 44. Normalmente, los servicios en un clúster de alta disponibilidad leen y escriben datos, por lo tanto, un clúster de alta disponibilidad debe mantener la integridad de los datos cuando un

nodo del clúster toma el control de un servicio de otro miembro del clúster. Las fallas de nodo en un clúster de alta disponibilidad no son visibles desde clientes fuera del clúster. Los clústeres de alta disponibilidad a veces se les denomina clústeres de conmutación por error. (Khedher, 2015)

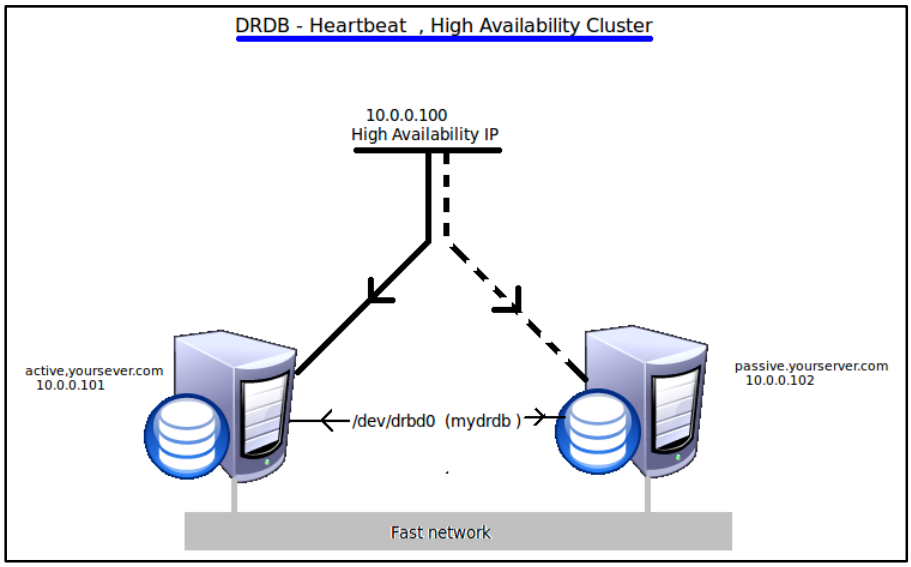


Figura 44. Clúster de Alta disponibilidad entre dos servidores.
Fuente: (Abdulkhareem, 2011).

Los clústeres de equilibrio de carga o cómputo adaptativo envían solicitudes de servicio de red a varios nodos del clúster para equilibrar el número de solicitudes entre nodos actuando como Front-end[8] del clúster. El equilibrio de carga proporciona una escalabilidad rentable ya que puede hacer coincidir la cantidad de nodos de acuerdo con los requisitos de carga. Si un nodo en un clúster de equilibrio deja de funcionar, el software detecta el error y redirige las solicitudes a otros nodos disponibles del clúster. Las fallas de nodo en un clúster de equilibrio de carga no son visibles desde clientes fuera del clúster, es necesario anclarse a este para tener acceso a los logs de error y notificaciones. La Figura 45 muestra el esquema de un clúster de balanceo de carga.

[8] Termino que se refiere a la separación de intereses entre una capa de presentación y una capa de acceso a datos. Se puede traducir como interfaz frontal o de interacción con el usuario.

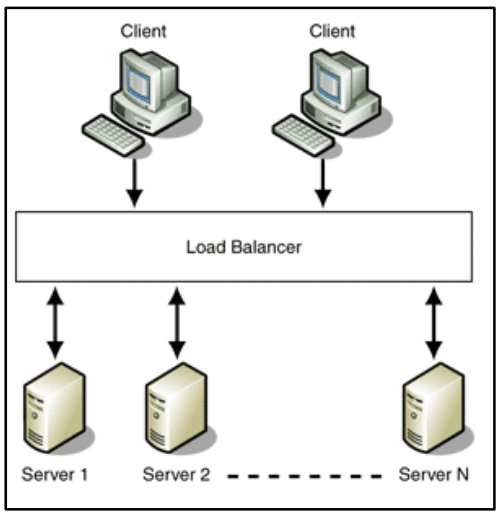


Figura 45. Componentes del balanceo de carga.
Fuente: (Microsoft Developer Network, 2003).

Los clústeres de alto rendimiento usan nodos de agrupamiento para realizar cálculos concurrentes. Un clúster de alto rendimiento permite que las aplicaciones funcionen en paralelo, mejorando así el rendimiento de las aplicaciones. Los clústeres de alto rendimiento también se conocen como clústeres computacionales o computación grid. Un ejemplo es el clúster computacional de Sum Microsystems ilustrado en la Figura 46.

Figura 46. Clúster Computacional de Sun Microsystems.
Fuente: (LIVERPOOL UNIVERSITY, 2014).

Los entornos de clúster también suelen utilizar un dispositivo de hardware externo que puede apagar una máquina que el clúster ya no puede alcanzar. Un dispositivo de corte garantiza la integridad de los recursos del sistema de archivos; si dos servidores accedieron a un sistema de archivos al mismo tiempo, podría dañar el sistema de archivos. A menudo se implementan tableros de administración remota, como HP Integrated Lights-Out o Dell Remote Access Controller, al igual que las unidades de distribución de energía externa como APC MasterSwitch o el interruptor de alimentación incluido en Cisco Unified Computing System. (Red Hat, Inc., 2007)

2.6.2. VENTAJAS DE UN CLÚSTER

La agrupación o clusterizacion es un mecanismo que permite que múltiples procesos y programas trabajen juntos como una sola entidad. Cuando se realiza búsquedas en la world wide web, puede parecer que la solicitud de búsqueda es procesada por un solo servidor web. En realidad, la solicitud de búsqueda es procesada por un gran grupo de servidores web conectados en clúster. Del mismo modo, se puede tener varias instancias de un mismo servicio trabajando juntas como una sola entidad para obtener ciertas ventajas como:

- Escalabilidad: si se tiene varias instancias de ejecución de un servicio, se puede hacer más trabajo y almacenar más datos de los que se podría con una sola instancia. También se puede dividir los datos en fragmentos más pequeños y distribuirlos en el clúster o realizar ciertas operaciones en ciertos miembros del clúster.

- Alta disponibilidad: Si se tiene varias instancias de ejecución de un servicio y una de ellas sufre un fallo, las otras instancias se mantendrán funcionando y disponibles.

- Persistencia de datos: No se perderá ningún dato almacenado en los servidores después de un reinicio manual, programado o un bloqueo. (OpenDaylight Project, 2016)

2.6.3. MÉTODOS DE DESPLIEGUE DE UN CLÚSTER

Al momento de realizar el despliegue de un clúster es necesario conocer cuáles son las variables del entorno, y la condiciones en las que se planea que el clúster funcione, así como la carga que se pretende inyectar en este. Por esta razón existen varios métodos de clusterizacion de servicios que permiten realizar una implementación a medida.

2.6.3.1. Clúster de Nodo Único

Los clústeres de nodo único son clúster de procesamiento con un solo nodo. Este nodo único actúa como maestro y esclavo para su clúster de proceso. Si bien los clústeres de un solo nodo solo poseen un miembro, la mayoría de los conceptos y funciones de procesamiento de información y operaciones de cálculo se siguen aplicando. Hay una serie de situaciones en las que los clústeres de procesamiento con un nodo único pueden ser útiles, como, por ejemplo:

- Probar versiones de software para computación grid y otros componentes de código abierto.
- Construcción de pruebas de concepto (PoC[9]).
- Ingeniería de datos de clase liviana.
- Procesamiento de datos no críticos a pequeña escala.
- Educación relacionada con ecosistemas de BigData.

[9] Implementación resumida o incompleta de, de un método o de una idea, realizada con el objetivo de verificar que una teoría tiene el potencial de ser explotada.

Como todo sistema los clústeres de nodo único tienen limitaciones relacionadas al procesamiento de información. Este tipo de clústeres no están destinados al procesamiento en paralelo a gran escala. En el caso de que se agoten los recursos, el clúster de nodo único requerirá el uso de elementos adicionales o un ambiente multinodo para suplir las deficiencias como ilustra la Figura 47.

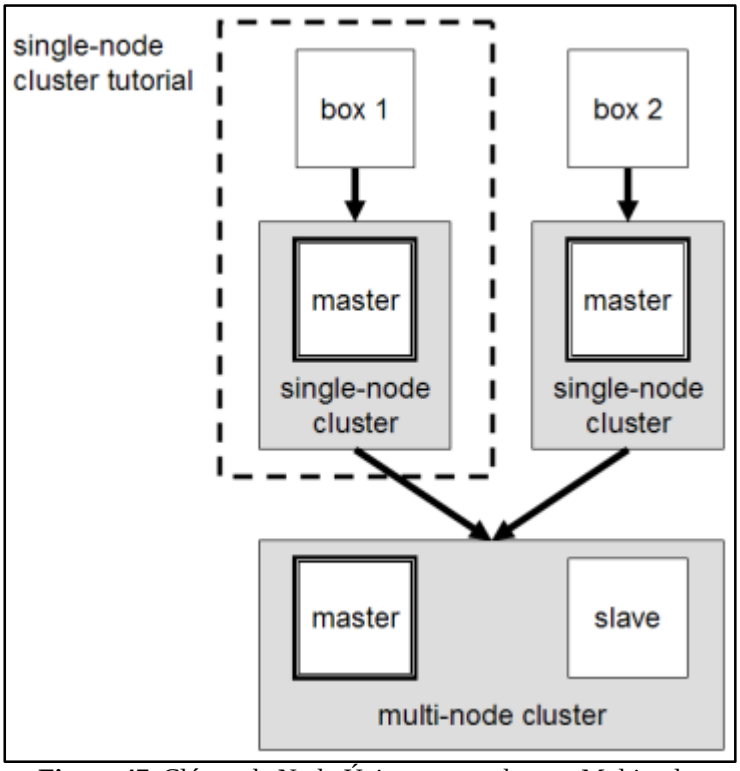


Figura 47. Clúster de Nodo Único conectado a un Multinodo.
Fuente: (Noll, 2011).

No se recomienda usar máquinas de tipo estándar con número reducido de núcleos debido a que los recursos son muy limitados y no permiten el procesamiento multi hilo. Los clústeres de nodo único no están adecuados con el soporte para alta disponibilidad porque solo poseen un miembro en el clúster y tampoco permiten el uso de máquinas virtuales con capacidades elevadas. (Google Developers, 2017)

2.6.3.2. Clúster Multi Nodo

La clusterizacion es una tecnología que permite proporcionar alta disponibilidad y rendimiento. Un clúster multi nodo consiste en la agrupación de dos o más servidores que ejecutan el mismo sistema operativo y el mismo servicio en donde existe más de un punto de gestión que controlan los nodos de esclavos. Después de consultar al nodo de administración, los nodos cliente se conectan directamente al servicio principal permitiendo reducir las latencias y la carga por el elevado número de solicitudes que se ejecutan concurrentemente.

Un clúster funciona mejor en entornos de nada compartido. Idealmente, no debería haber dos componentes que compartan el mismo hardware. Por motivos de simplicidad se suele usar dos nodos de control sincronizados entre ellos y a partir del tercero se ejecutan las operaciones de administración del clúster y procesamiento de información como ilustra la Figura 48.

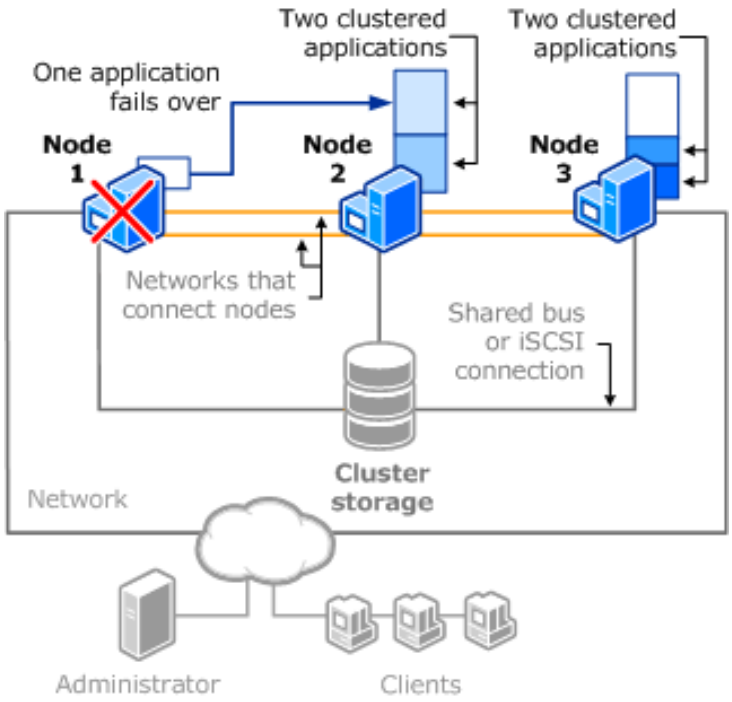


Figura 48. Fail-Over en un clúster multinodo.
Fuente: (Microsoft, 2008).

Para implementar un clúster de múltiples nodos se toman algunas consideraciones como el uso mínimo de tres servidores para el alojamiento de los servicios principales que conforman el clúster. Es posible crear un clúster con solo dos nodos, sin embargo, si uno de los dos entra en estado de falla, el clúster dejara de estar operativo. Esto se debe a que la agrupación de servicios requiere que la mayoría de nodos se encuentren activos y que un nodo no puede ser la mayoría de dos nodos.

Todos los dispositivos que pertenecen a un clúster deben tener un identificador. Para esto se usa el rol del nodo que permite establecer el orden de los miembros y la jerarquía dentro del clúster. Si se tiene un clúster de tres nodos y se desea tolerancia a bloqueos con un solo nodo, debe ejecutarse una réplica de cada fragmento de datos de identificación en los tres nodos del clúster.

CAPITULO III

3. DESARROLLO DEL PROYECTO

En este capítulo se detalla el proceso llevado a cabo para la consecución satisfactoria del despliegue. Entre los temas principales que se abordan en esta sección se encuentra el análisis de la situación actual de la red del centro de datos de la Facultad de Ingeniería en Ciencias Aplicadas, así como las condiciones de su nube privada sobre la cual se desarrollara el presente proyecto. Se tratan también temas como la selección del nuevo controlador de red SDN, de acuerdo a los parámetros dictados por el estándar ISO/IEC/IEEE 29148 además de la adecuación respectiva del entorno cloud para que este sea capaz de comunicarse con el controlador de red externo. Las normas que se usan en este apartado se encuentran en el ANEXO A-1.

3.1. SITUACIÓN ACTUAL

La Facultad de Ingeniería en Ciencias Aplicadas cuenta con un centro de datos TIER I totalmente funcional, encargado de albergar una serie de servicios locales. La estructura de la red de la Universidad Técnica del Norte y del centro de datos FICA se muestra en la Figura 49, en donde se puede apreciar cual es la distribución actual y sus respectivas etiquetas.

En el área correspondiente al equipamiento del centro de datos FICA está situado un switch de distribución, que se encuentra a su vez conectado a otro switch no administrable de la marca Linksys, que sirve de puente a los servidores de la sección de cloud y a los equipos que brindan servicios locales como Opina, Reactivos, Geoportal y DSpace. Cada uno de estos dispositivos tienen asignada una dirección IP perteneciente al pool de la zona desmilitarizada (DMZ) de la Universidad Técnica del Norte y cuentan también con una dirección de acceso local generada a partir del pool de direcciones de la red privada de la institución.

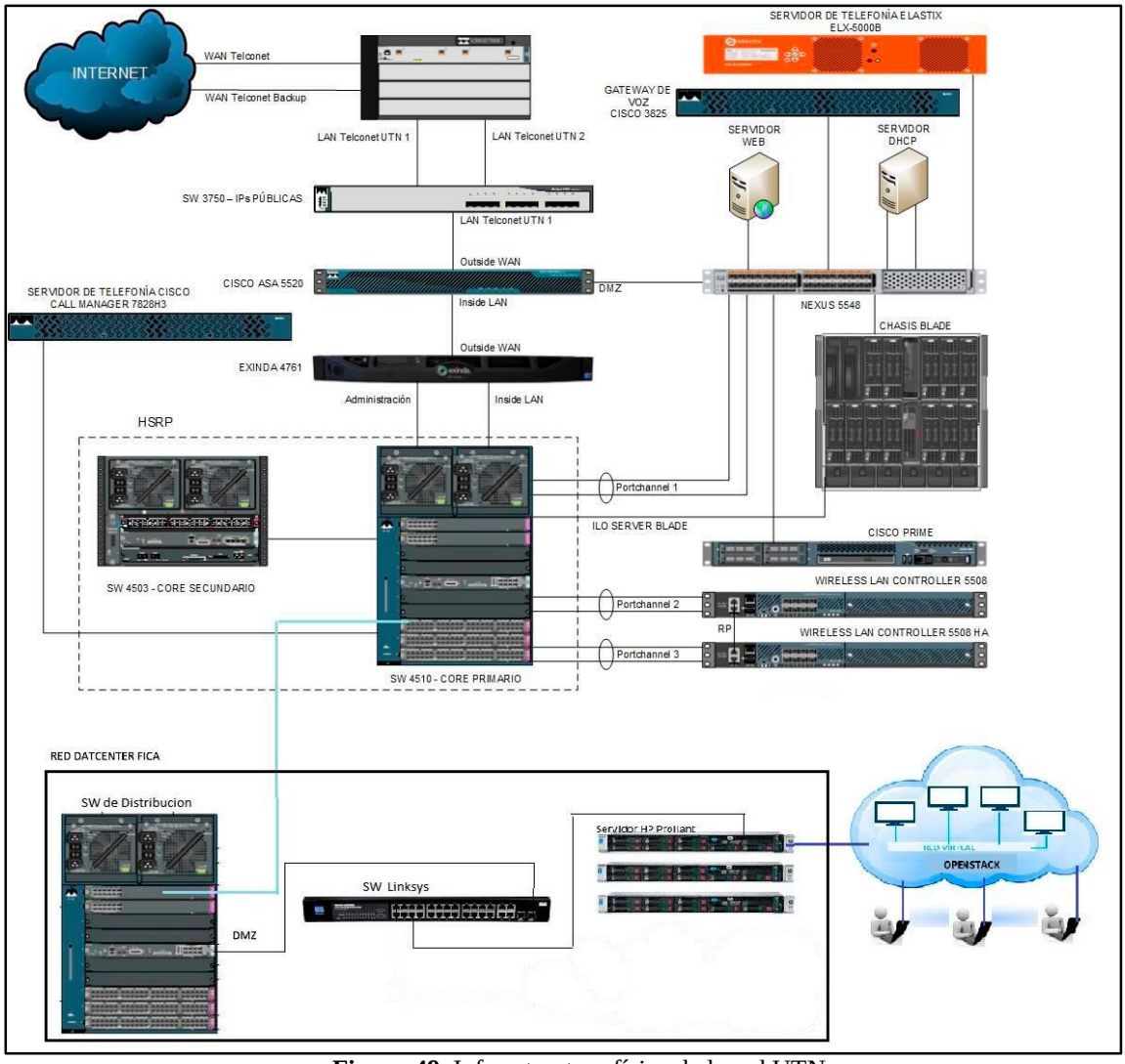


Figura 49. Infraestructura física de la red UTN.
Fuente: Dirección de Desarrollo Tecnológico e Informático UTN.

El presente proyecto incluye algunos componentes nuevos que no alteran la distribución de equipos del centro de datos FICA, todos estos elementos se conectaran a través del mismo puente de red o switch denominado Linksys haciendo que el despliegue sea transparente. El acceso a internet se adquirirá por medio del mismo direccionamiento de la red de servidores de la UTN, permitiendo que los recursos de la nube local y el controlador de red SDN se vuelvan parte de la red institucional.

3.1.1. INFRAESTRUCTURA FISICA Y LÓGICA DEL CENTRO DE DATOS FICA

El centro de datos FICA está constituido por elementos de hardware de propósito específico, como servidores de chasis, servidores de rack, switches administrables, de core y distribución. Cada uno de estos elementos cuenta con su respectivo cableado hacia los componentes que proporcionan conectividad a través del centro de datos principal en la Dirección de Desarrollo Tecnológico e Informático de la Universidad Técnica de Norte. El software utilizado en los servidores en su mayoría es de código abierto por lo que no son necesarias licencias para su funcionamiento. Las Tabla 4 y 5 detallan las características y las funciones de cada uno de los switches y servidores del centro de datos FICA.

Tabla 4.

Características de los switches del centro de datos FICA

	FUNCIÓN	**CARACTERÍSTICAS**			**NOMBRE**
		PUERTOS	**MODELO**	**MARCA**	
	Enlace principal. Distribución de la red FICA	48 Gbps / all slots 24 Gbps/slot (slot 1-7) 6 Gbps/slot (slot 8-10)	Catalys 4506-E	Cisco	Switch de Core
	Acceso a la red	24-port 10/100 + 4-Port Gigabit	SR224G	Linksys	Switch de Distribución

Tabla 5.

Características de los servidores de centro de datos FICA.

	FUNCIÓN	CARACTERÍSTICAS					MODELO DEL SERVIDOR
		SISTEMA OPERATIVO	NIC DE RED	DISCO DURO	PROCESADOR	MEMORIA RAM	
	DHCP FICA	Ubuntu Server 14.04 LTS	1 Gigabit Ethernet Broadcom BCM5722	1 Tb	Dual-core Intel Xeon E3110 3.0 GHz	2 Gb	IBM System x3200 M2
	OpenStack Cloud Server	Ubuntu Server 14.04 LTS	4 x 1 GbE FLR-T FlexFabric Flexible LOM	3x450 GB	1x Intel Xeon E5 2620 V3	32 GB	HP DL360 G9
	Eucalyptus Cloud	CentOS 6.5	4 x 1 GbE FLR-T FlexFabric Flexible LOM	3x450 GB	1x Intel Xeon E5 2620 V3	32 GB	HP DL360 G9
	Open Nébula Cloud Server	CentOS 7	4 x 1 GbE FLR-T FlexFabric Flexible LOM	3x450 GB	1x Intel Xeon E5 2620 V3	32 GB	HP DL360 G9
	Red Virtual SDN	Ubuntu Server 14.04 LTS	1 Gigabit Ethernet HP NC105	160 GB	Intel Xeon E5405 Quad Core	1 GB / 16 GB (max.)	HP ProLiant ML150 G5
	Encuestas Opina	Ubuntu Server 12.04 LTS	1 Gigabit Ethernet HP NC105	160 GB	Intel Xeon E5405 Quad Core	1 GB / 16 GB (max.)	HP ProLiant ML150 G5

3.1.2. ESTADO DEL CLOUD FICA

La primera etapa del desarrollo del proyecto consiste en el levantamiento de información en el centro de datos FICA. Mediante este proceso se recopilan los datos de la situación actual del sistema, permitiendo identificar si existen problemas y cuál es el grado de mejora que el proyecto ofrecerá a su término.

Durante la primera inspección realizada al centro de datos se dio lugar a la entrega/recepción de los accesos a las plataformas cloud activas que son objeto del análisis de compatibilidad con la integración de SDN. OpenNebula, Eucalyptus y OpenStack son los entornos que se encuentran a disposición del presente proyecto. Y según la cantidad de documentación existente la plataforma más robusta de las tres estudiadas es OpenStack, por lo que se toma en cuenta para el despliegue de este proyecto. La Figura 50 muestra el dashboard de OpenStack Cloud Server en el centro de datos FICA.

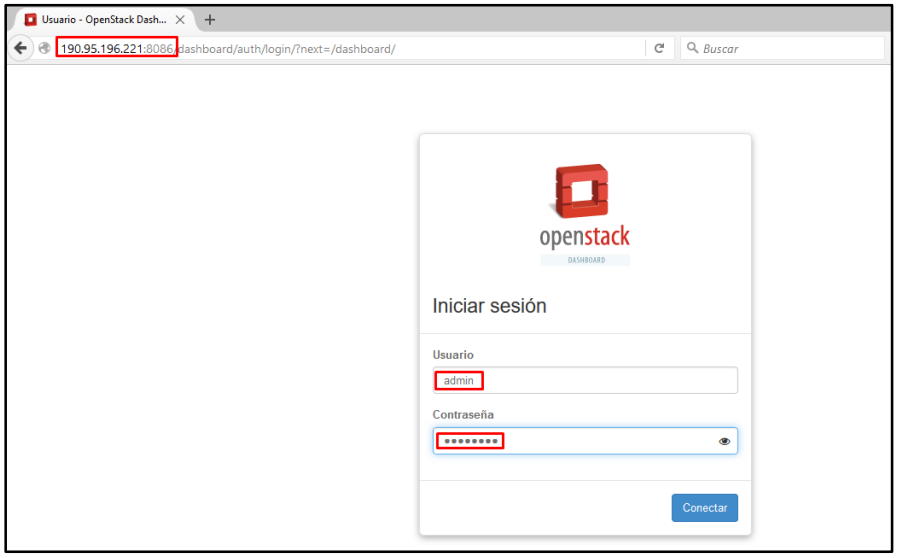


Figura 50. Dashboard OpenStack FICA.

3.1.3. APLICACIÓN DEL MÉTODO IQMC PARA VALIDAR EL CLOUD FICA

El despliegue de SDN integrado al cloud requiere que las características de la plataforma de nube sean evaluadas conscientemente para que el resultado sea satisfactorio. Es por eso que se ejecuta un proceso para la creación de una matriz de evaluación de plataformas de código abierto, basándose en los pasos del método IQMC[10] el cual proporciona las guías y técnicas para identificar los factores de calidad apropiados de los componentes de software. Los pasos del método IQMC se muestran en la Figura 51. (Bermeo, Sanchez, Maldonado, & Carvallo, 2005)

Mediante la aplicación de este método el resultado debe cumplir con cuatro principios:

- Debe poseer características de calidad de alto nivel.
- Debe permitir jerarquía en las características de calidad para estructurar el modelo.
- Debe permitir encubrimiento para evitar omitir características dependientes.
- Debe ser general para evitar características no útiles para la ingeniería de software.

[10] Método que brinda directrices y técnicas para definir modelos de calidad de diversos tipos de software según la estructura del estándar ISO 25000.

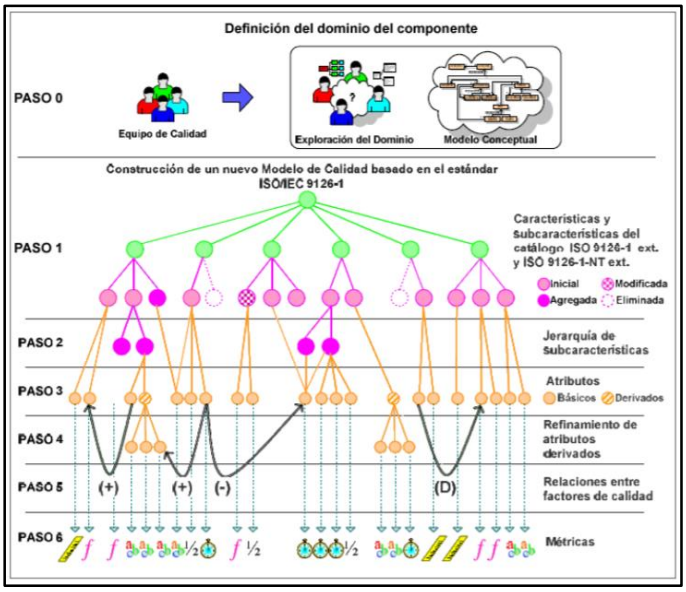


Figura 51. Método IQMC.
Fuente: (Bermeo, Sanchez, Maldonado, & Carvallo, 2005).

3.1.4. CONSTRUCCIÓN DEL MODELO DE CALIDAD

En esta sección se describe el proceso para la creación de una matriz de evaluación para las plataformas de código abierto activas en el centro de datos FICA y que son candidatas a ser usadas como framework para el despliegue del proyecto SDN, tomando en cuenta los pasos que establece el método IQMC y los factores de calidad que dicta el estándar ISO/IEC 25000[11].

Paso 0: Estudio del Dominio.

Del análisis comparativo se seleccionará la plataforma que cuente con las mejores características, para proceder al despliegue e implementación de los complementos de SDN sobre el servicio de nube computacional de tipo IaaS para proporcionar los servicios:

[11] Familia de normas para la creación de un marco de trabajo común para evaluar la calidad del producto software.

- Escritorios Virtuales
- Volúmenes de Almacenamiento
- Procesamiento Paralelo
- Balanceo de Carga

Paso 1 y 2: Determinación de características de calidad.

Una vez ejecutado el paso cero del modelo IQMC y fundamentándose en el estándar ISO/IEC 25000, se establecen las características y sub características que serán evaluadas de acuerdo a las necesidades que el centro de datos FICA presenta en el momento de realizar el despliegue del presente proyecto. La Tabla 6 detalla las características y sub características establecidas.

Tabla 6.

Características y Subcaracterísticas según ISO/IEC 25000.

	SUBCARACTERISTICAS			
CARACTERÍSTICAS	Funcionalidad	Adecuación	Exactitud	Seguridad de acceso
	Fiabilidad	Madurez	Tolerancia a fallos	Recuperabilidad
	Portabilidad	Adaptabilidad	Coexistencia	
	Mantenibilidad	Capacidad para ser probado	Capacidad para ser analizado	Estabilidad
	Eficiencia	Comportamiento en el tiempo	Utilización de recursos	
	Usabilidad	Capacidad para ser aprendido	Capacidad para ser operado	Análisis de documentación.
	Seguridad	Vulnerabilidad		
	Satisfacción	Cuestionario de Satisfacción		
	Interoperabilidad	Interoperabilidad		

Paso 3 y 4: Refinamiento de Subcaracterísticas en atributos derivados.

Con la definición concreta de las características y Subcaracteristicas que se van a evaluar se procede a descomponer cada subcaracteristica en atributos derivados para realizar la medición de una manera mucho más directa. En la Figura 52 se indica la descomposición realizada a una de las subcaracterísticas.

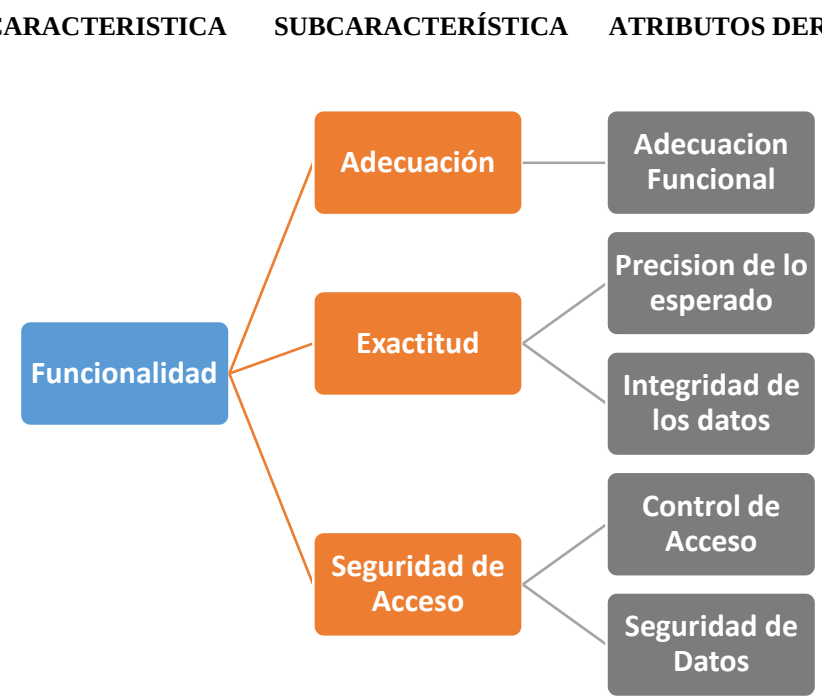


Figura 52. Descomposición de una característica.

Paso 5: Establecimiento de relaciones entre factores de calidad.

Además de realizarse la pura descomposición jerárquica de cada una de las características y subcaracterísticas para obtener los atributos derivados que suministran información más detallada, los factores de calidad se relacionan por otros criterios. La Figura 53 indica las relaciones entre factores de calidad que se usan para el desarrollo del modelo.

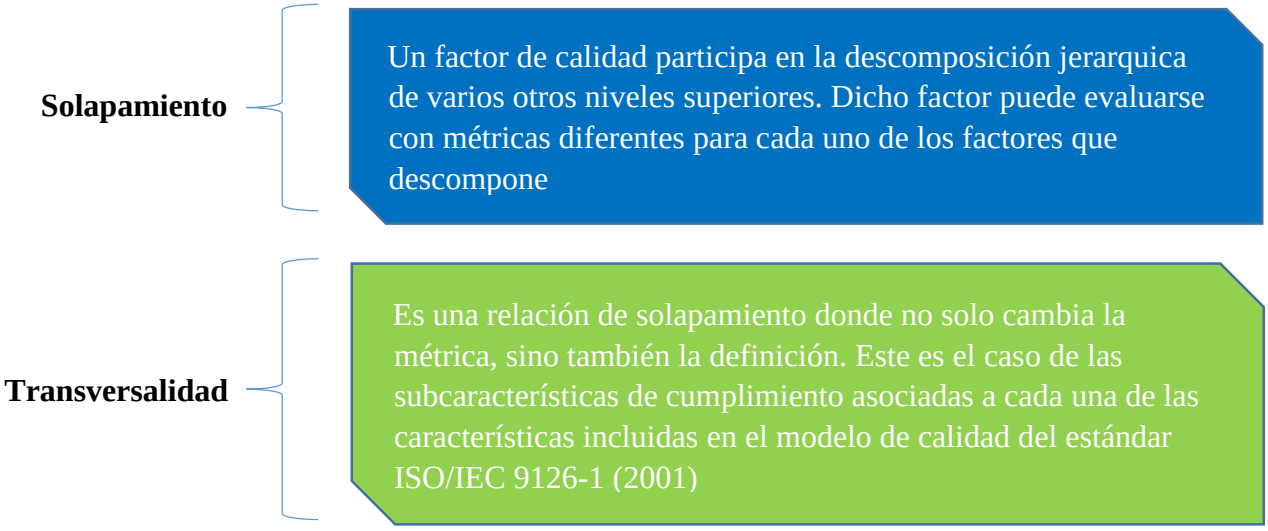


Figura 53. Relaciones entre factores de calidad.
Fuente: (Bermeo, Sanchez, Maldonado, & Carvallo, 2005).

Paso 6: Determinación de métricas para los atributos.

En este paso se establecen las métricas que se ocuparan para realizar la evaluación de las características, sub características y sus respectivos atributos en base a la norma ISO/IEC 25000. Cada métrica cuenta con un equivalente contable que puede ser uno o cero, de esta manera al contabilizar los registros en la matriz se obtendrán valores en porcentaje representativos. La Tabla 7 describe las métricas usadas para evaluar el cumplimiento de las características y la Tabla 8 describe las métricas por rango de intervalos enteros.

Tabla 7.

Métrica de cumplimiento de características

Nro.	Cumplimiento	Equivalente Contable
1	Si	1
2	No	0

Tabla 8.

Métrica de cumplimiento por rango.

Nro.	Cumplimiento	Equivalente Contable
1	Nulo	0
2	Bajo	1
3	Medio	2
4	Alto	3
5	Completo	4

3.1.5. RESULTADO DE LA EVALUACIÓN

Después de construir el modelo de evaluación y generar las matrices del ANEXO B con los parámetros establecidos por la norma ISO/IEC 25000, se aplican las métricas a los atributos y se realiza la suma de los puntajes asignados a cada una de las características de calidad; obteniendo los resultados que se detallan en la Tabla 9. El porcentaje evaluado de cada característica se obtiene en base a los parámetros de División de Evaluación de Calidad que se hallan descritos en la norma ISO/IEC 2504n. Esta norma se encuentra en el ANEXO A-2.

Tabla 9.

Tabla de resultados del análisis comparativo de plataformas cloud.

Parámetros Técnicos ISO 25000	%	Puntos	OpenStack		OpenNebula		Eucalyptus	
			Puntos	%	Puntos	%	Puntos	%
Funcionalidad	20	12	20	12	18	11	20	12
Fiabilidad	15	16	17	16	13	14	12	13
Portabilidad	10	4	2	4	10	4	10	4
Mantenibilidad	10	15	20	13	8	12	7	10
Eficiencia	10	13	14	11	5	7	5	7
Usabilidad	15	12	9	11	8	6	8	6
Seguridad	15	3	1	3	5	1	5	1
Interoperabilidad	5	2	1	2	5	2	5	2
TOTAL	**100**	**77**	**83**	**72**	**72**	**57**	**72**	**55**

De acuerdo a los resultados obtenidos al momento de aplicar la matriz comparativa entre las plataformas OpenStack, OpenNebula y Eucalyptus, el resumen de resultados nos advierte que OpenStack tiene un nivel superior respecto a las otras dos plataformas evaluadas, por lo que se ha determinado el uso de OpenStack como marco de trabajo para el despliegue de la red SDN. La Figura 54 muestra un esquema grafico de los resultados obtenidos en la evaluación comparativa de las plataformas OpenStack, OpenNebula y Eucalyptus.

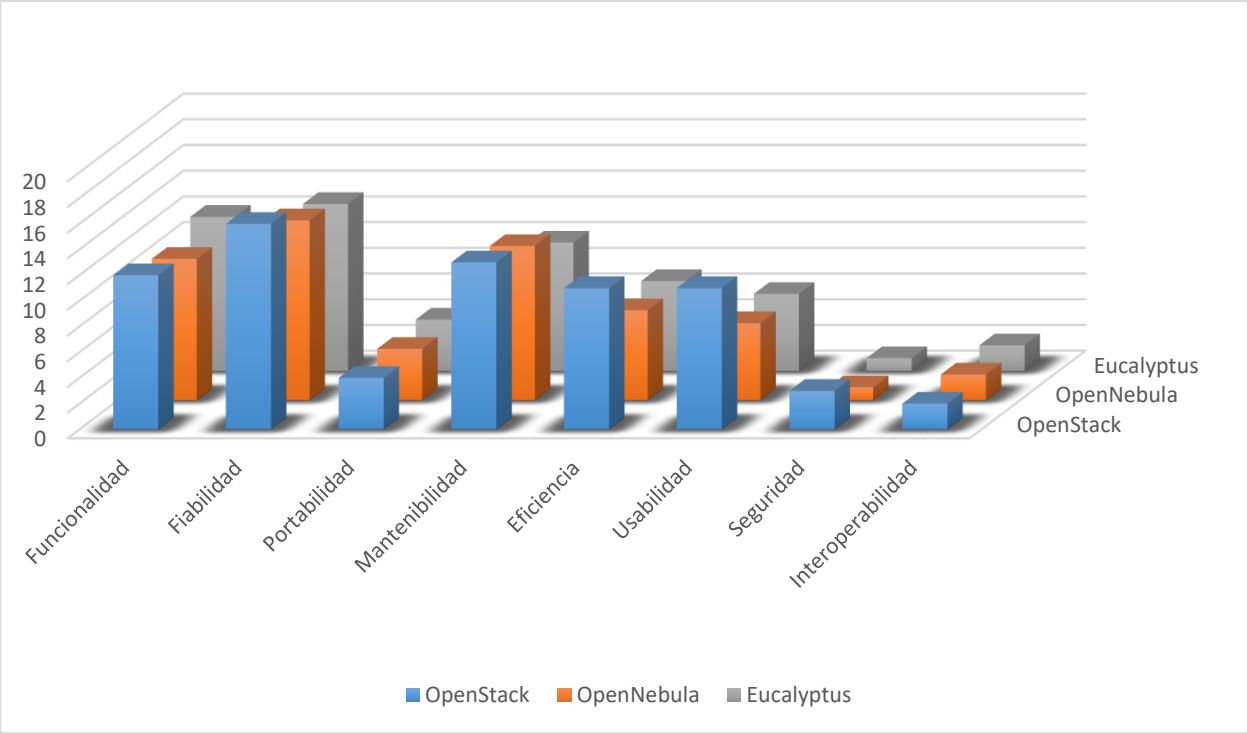


Figura 54. Esquema gráfico del análisis comparativo de las plataformas.

3.2. ACONDICIONAMIENTO DE LA PLATAFORMA CLOUD

OpenStack es una plataforma que funciona a base de múltiples servicios integrados, pero tres de sus componentes principales se encargan de las funciones de computación, redes y almacenamiento. Dichos componentes se comunican entre sí, bajo protocolos de estándares

abiertos. Los usuarios tienen acceso a la configuración a través de la interfaz web Horizon con la que se pueden delegar operaciones a perfiles protegidos por usuario y contraseña.

Debido a la naturaleza modular de OpenStack, la integración de elementos adicionales requiere que se realicen adecuaciones o configuraciones en los demonios de servicio para que el funcionamiento correcto no se vea afectado después de agregar agentes de terceros. La ventaja que posee OpenStack es su intuitivo método de combinación a base de librerías escritas para propósitos específicos.

Según el resumen de la plataforma mostrado en la Figura 55, la cantidad de recursos asignados es ineficiente e insuficiente, debido a que, el modo de operación establecido durante la instalación del software base es todo en uno. Por esta razón es necesario realizar la reconstrucción de la plataforma usando un modelo más robusto como el multi nodo aplicando clusterización y con sus servicios distribuidos, así se asegura que los recursos sean asignados adecuadamente y a la vez se permite la integración de nuevos componentes de manera más flexible.

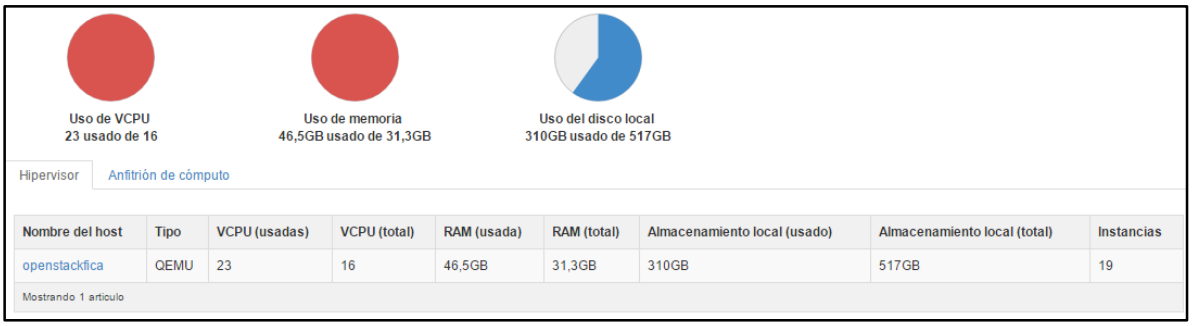


Figura 55. Recursos asignados en OpenStack.
Fuente: OpenStack Cloud Server Centro de Datos FICA.

Al momento de realizar este despliegue no existen instancias funcionales sobre OpenStack por lo que no hay riesgo de pérdida de información. En caso de existir alguna instancia corriendo sobre

la plataforma se debe recurrir a las funciones de respaldo o migración en vivo que proporcionan los gestores de infraestructura.

3.2.1. INSTALACIÓN DE PRE-REQUISITOS DEL ENTORNO DE VIRTUAL

La segunda etapa consiste en la reconstrucción de la plataforma cloud para que se adecue a las necesidades de SDN. Esto se logra reinstalando y reconfigurando los módulos de cómputo y de red para que trabajen paralelamente con el nodo de control de red que se despliega en una sección siguiente. Debemos asegurarnos de que el sistema operativo se encuentre totalmente actualizado para obtener los paquetes más recientes del gestor de infraestructura.

Para realizar la actualización del sistema debemos acceder a través de una sesión SSH usando la dirección IP del servidor. Para efectos de explicación del proceso, accedemos usando la dirección original dentro del centro de datos y cuando nos pidan el usuario y el password ingresamos las credenciales configuradas al momento de la instalación del sistema operativo del servidor y ejecutamos el comando *yum update* para que se inicie una búsqueda de actualizaciones y cuando termine podemos reiniciar el sistema para estabilizar los paquetes. El proceso llevado a cabo satisfactoriamente se muestra en la Figura 56.

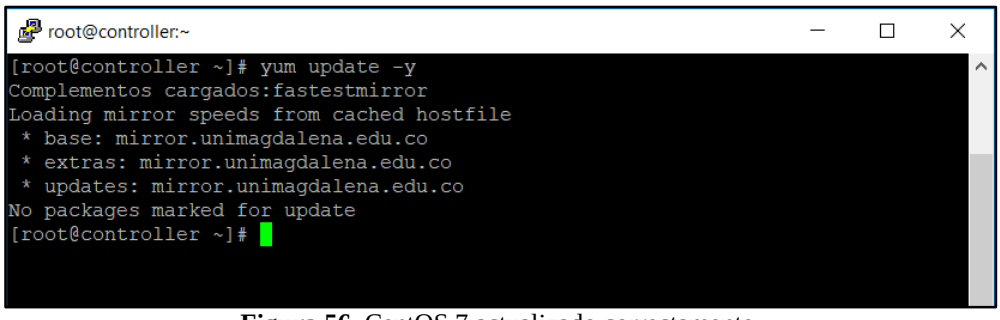


Figura 56. CentOS 7 actualizado correctamente.

Uno de los requisitos fundamentales para la correcta implementación de cualquier plataforma basada en Linux es la correcta configuración del fichero hosts en donde se encuentran los punteros a los dominios y servicios locales más importantes. Para ello vamos a editar el fichero que se encuentra en la dirección /etc/hosts y agregamos las entradas correspondientes a los nodos que serán parte del entorno virtualizado tal como se observa en la Figura 57.

- Controller 10.24.8.74
- Compute1 10.24.8.75
- Compute2 10.24.8.76
- SDN 10.24.8.77

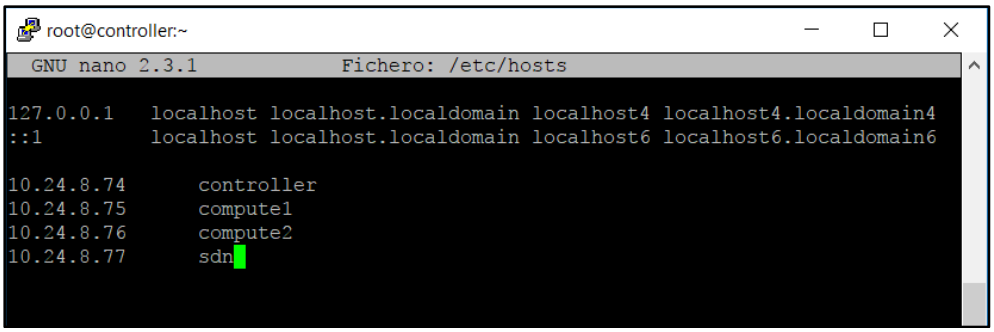


Figura 57. Configuración de entradas en el fichero hosts.

El siguiente paso es deshabilitar los mecanismos de firewall del sistema operativo y para ello ejecutamos los comandos tipeados en la consola de la Figura 58 que aseguran que no existan bloqueos nuestras conexiones durante la etapa de configuración. También deshabilitaremos SELinux y el administrador de redes para evitar que la protección de Linux banee los servicios que necesitan alteraciones en los ficheros de configuración y ejecución.

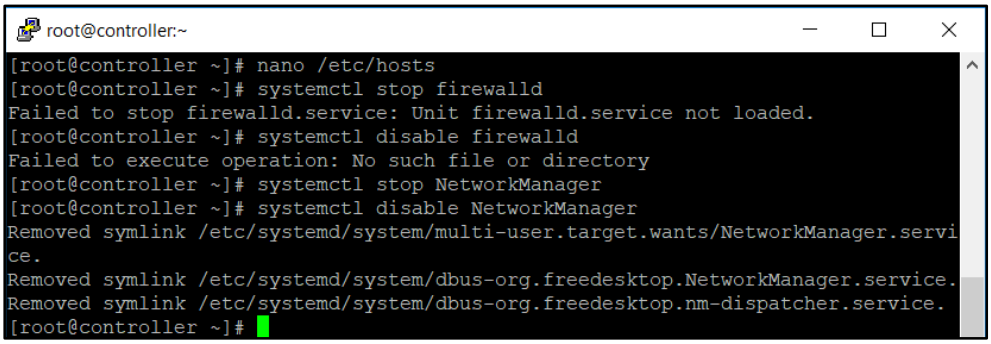


Figura 58. Desactivación de los mecanismos de seguridad de Linux.

Ahora generamos una clave SSH con el comando de la Figura 59 para permitir el acceso sin contraseña a los nodos de confianza de la SDN. Con esto los comandos se ejecutan sin necesidad de escribir la contraseña de root cuando es necesario hacer cambios en ficheros del sistema. Esto se debe realizar con sumo cuidado, de no dejar vulnerable al nodo principal de manera equivocada. Este paso se realiza también durante la clusterización.

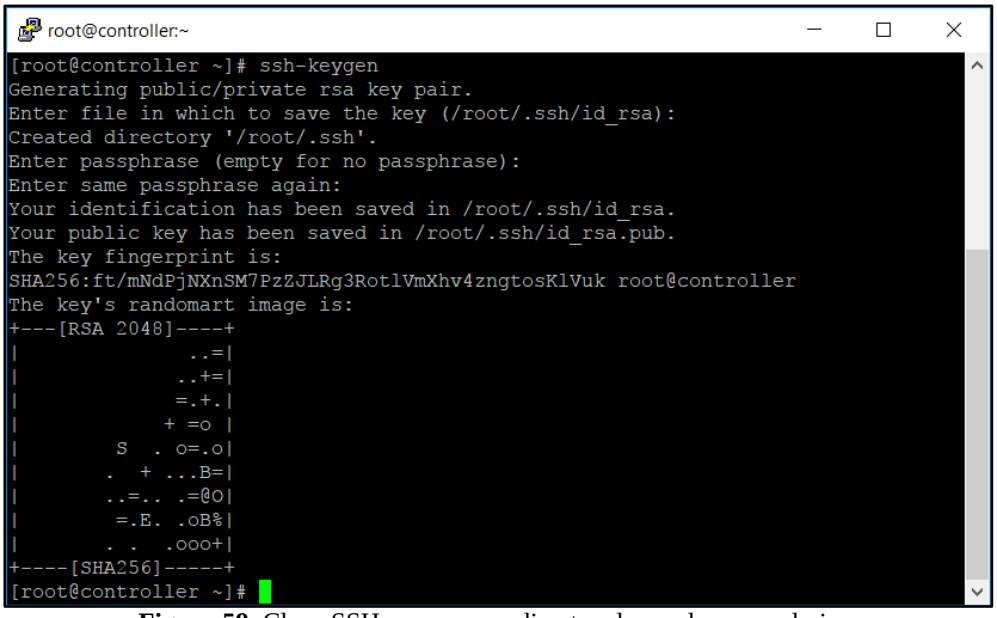


Figura 59. Clave SSH para acceso directo a los nodos secundarios.

3.2.2. CONFIGURACIÓN DE SERVICIOS PARA OPENSTACK

OpenStack es un framework constituido por diversos componentes que desempeñan funciones necesarias para el funcionamiento de una nube computacional, ya sea pública o privada. Todos los elementos combinados y bien configurados son capaces de ofrecer un servicio IaaS listo para ser consumido. Una de las características de OpenStack es la posibilidad de utilizar diferentes herramientas de despliegue o instalación como PackStack, que permite manipular la instalación de casi todos sus componentes a pequeña o gran escala. Por esta razón se usa el método de fichero de respuestas para crear el ambiente y luego automatizar la instalación con PackStack.

Lo primero que se debe hacer es descargar los repositorios más actuales de OpenStack e instalar la herramienta PackStack como indica la Figura 60. Una vez que se han realizado esto se procede a generar el fichero de respuestas como se muestra en la Figura 61 y, a partir de este fichero se crea el ambiente a desplegar modificándolo de acuerdo al tipo de uso que se dará a la plataforma.

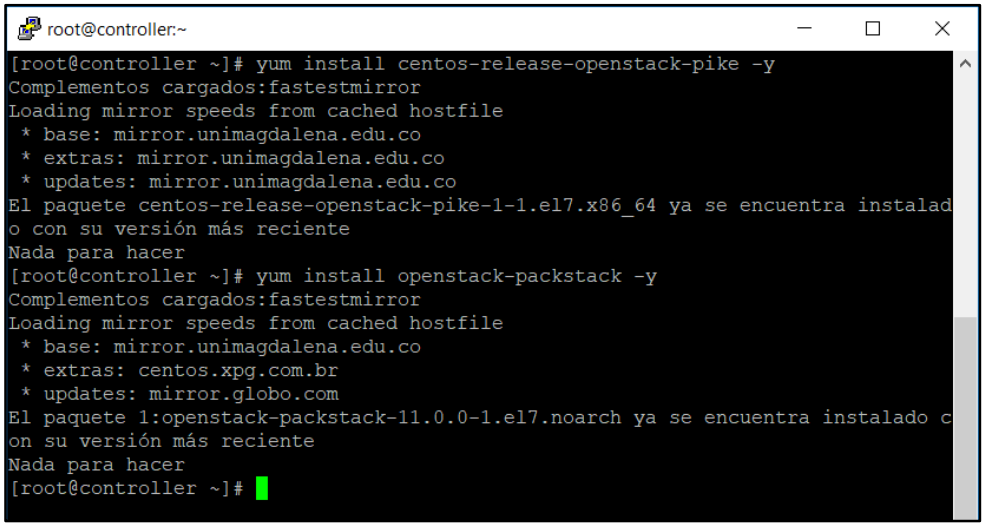


Figura 60. Instalación del repositorio de OpenStack.

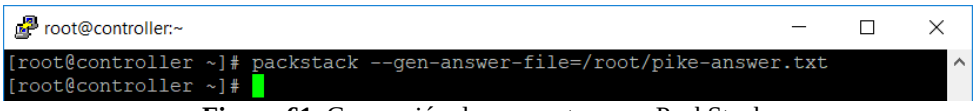


Figura 61. Generación de respuestas para PackStack.

Dentro del fichero de respuestas de la Figura 62, existe una gran variedad de apartados, pero los más relevantes para este proyecto son los relacionados con volúmenes de almacenamiento, procesamiento paralelo y redes. Debido a esto se desactivan los servicios no críticos y se dejan únicamente los básicos para que la nube funcione durante las pruebas del entorno SDN. Los cambios en la arquitectura se efectúan mediante la modificación de los siguientes campos:

- *CONFIG_CONTROLLER_HOST=10.24.8.74*
- *CONFIG_COMPUTE_HOSTS=10.24.8.74*
- *CONFIG_NETWORK_HOSTS=10.24.8.74*
- *CONFIG_PROVISION_DEMO=n*
- *CONFIG_CEILOMETER_INSTALL=n*
- *CONFIG_NEUTRON_LBAAS=y*
- *CONFIG_NEUTRON_FWAAS=y*
- *CONFIG_NEUTRON_VPNAAS=y*
- *CONFIG_NEUTRON_OVS_BRIDGE_MAPPINGS=extnet:br-ex*
- *CONFIG_NEUTRON_OVS_BRIDGE_IFACES=br-ex:ens33*

- *CONFIG_NEUTRON_OVS_EXTERNAL_PHYSNET=extnet*

- *CONFIG_KEYSTONE_ADMIN_PW=cloudfica2017,.*

Una vez que hemos modificado los campos necesarios y más importantes procedemos a ejecutar el despliegue mediante la herramienta PackStack la cual tomara algún tiempo en elaborar la plataforma dependiendo de la capacidad de procesamiento del servidor y de la conexión a internet para descargar los paquetes necesarios del repositorio de OpenStack.

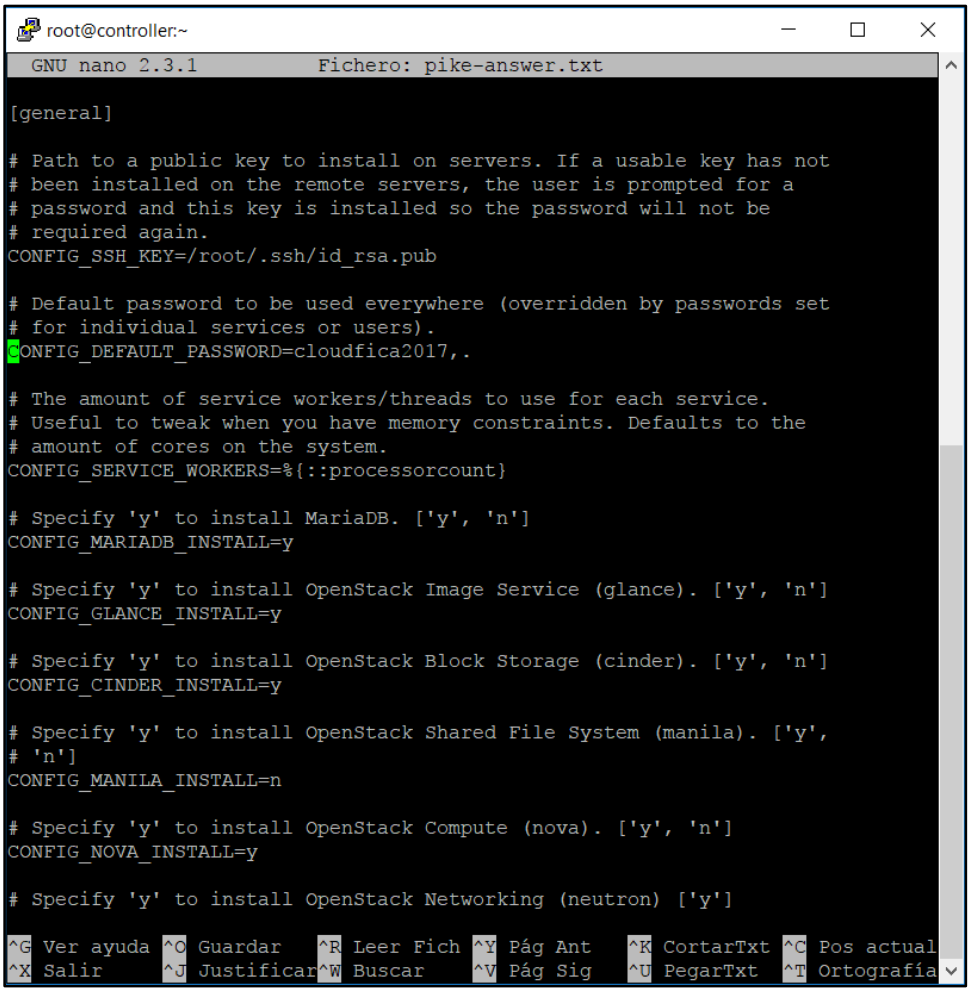


Figura 62. Fichero de respuestas de PackStack.

PackStack se encarga de comprobar las configuraciones previas de nuestro equipo para indicar mediante notificaciones si es posible instalar los paquetes en el servidor o si existe algún problema de configuración. La mayoría de veces los errores se presentan por ficheros desconfigurados o por bloqueos en la ejecución provenientes de SELinux. De no existir ningún inconveniente se verá una imagen como la de la Figura 63.

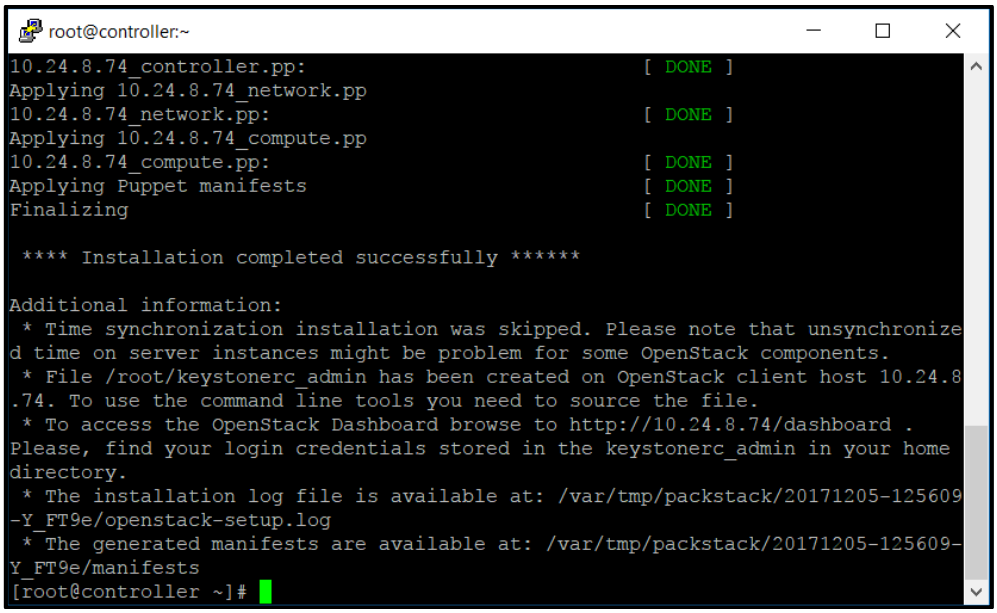


Figura 63. Instalación Exitosa de OpenStack.

Ahora se realiza una comprobación de que la plataforma se está ejecutando correctamente mediante el acceso a través de su dirección IP estática en el navegador. La primera vez puede tardar unos segundos debido a que el servicio web se encuentra estabilizándose. A partir de aquí se debe observar una imagen similar a la que muestra la Figura 64, en donde escribiremos el usuario y la contraseña de administrador ara acceder al panel de administración de la plataforma.

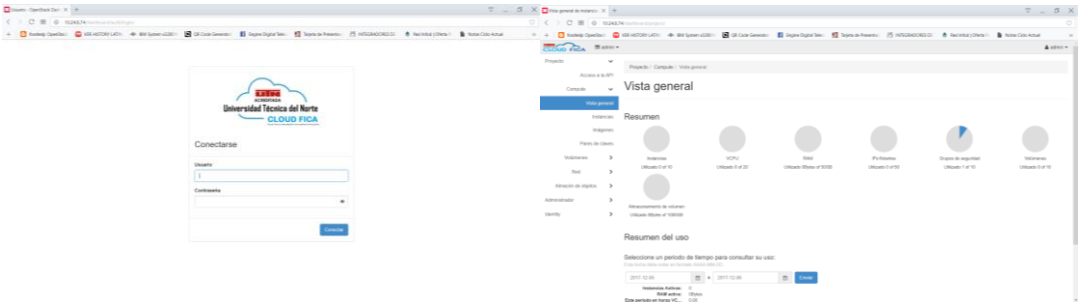


Figura 64. Dashboard de OpenStack.

Para terminar con el acondicionamiento de la plataforma cloud, se verifica el estado de los servicios Keystone (Identity), Glance (Image Storage) y Nova (Compute), mediante la creación de un flavor, una red y una instancia de prueba mediante el gestor de infraestructura como ilustra la Figura 65. Por el momento la red externa es inaccesible ya que el módulo de red no se encuentra correctamente configurado. El proceso de creación de instancias se detalla en el manual de operación de OpenStack del ANEXO C.

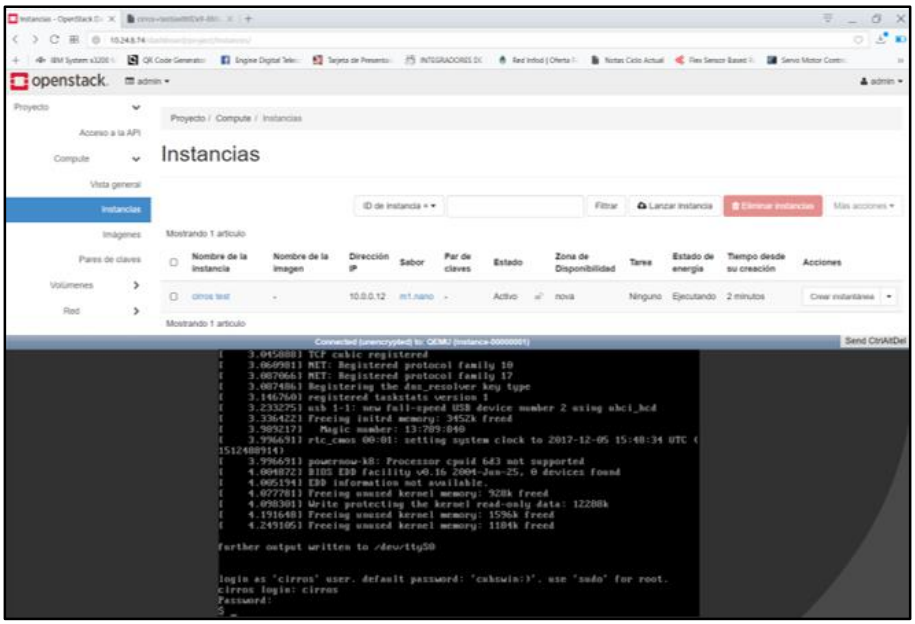


Figura 65. Instancia funcionando sobre OpenStack.

3.2.3. CONFIGURACIÓN DEL SERVICIO DE NETWORKING – NEUTRON

El módulo de Networking de OpenStack permite crear y administrar objetos de red, como subredes y puertos, que otros servicios pueden usar. Los complementos que se pueden implementar para admitir diferentes equipos de red y software, proporcionan flexibilidad a la arquitectura de la nube de OpenStack. (OpenStack, 2016)

En este apartado se configura el módulo Neutron para realizar la implementación de un escenario clásico del servicio OpenStack Networking usando el complemento ML2 con Open vSwitch. Esta implementación proporciona un método para que los usuarios sin privilegios puedan tener acceso a las redes virtuales dentro de un proyecto e incluye los siguientes componentes:

- Redes de proyecto (tenant[12]): proporcionan conectividad a instancias para un proyecto en particular. Los usuarios sin privilegios pueden administrar redes de proyecto dentro de la asignación que el administrador asigne para ellos. Las redes de proyecto pueden usar los métodos de transporte VLAN, GRE o VXLAN.

- Redes externas: brindan conectividad a redes externas como internet. Solo los usuarios administrativos pueden administrar las redes externas porque interactúan con la infraestructura de red física. Las redes externas pueden usar métodos de transporte planos o de VLAN, dependiendo de la infraestructura física de la red y generalmente usan rangos de direcciones IP públicas.

El proceso para habilitar Neutron en OpenStack y habilitar el acceso de las instancias a la red externa puede ser bastante largo. El uso de PackStack para el despliegue del ambiente cloud facilita

[12] Conjunto de recursos de computación, almacenamiento y red asignados a un grupo aislado de usuarios.

en cierta medida la configuración del módulo de networking ya que instala una plantilla pre configurada que se debe adecuar mediante la modificación de los ficheros de interfaces del servidor.

Cuando se despliega OpenStack mediante PackStack, se crea una interfaz bridge (br-ex) en el nodo de red, que en este caso es el mismo nodo controlador. Para habilitar la interfaz bridge es necesario editar el fichero de la interfaz ethernet en /etc/sysconfig/network-scripts/ y agregar un nuevo fichero para br-ex donde se añade la interfaz ethernet como un puerto y se asigna la dirección IP de la interfaz ethernet como se indica en la Figura 66.

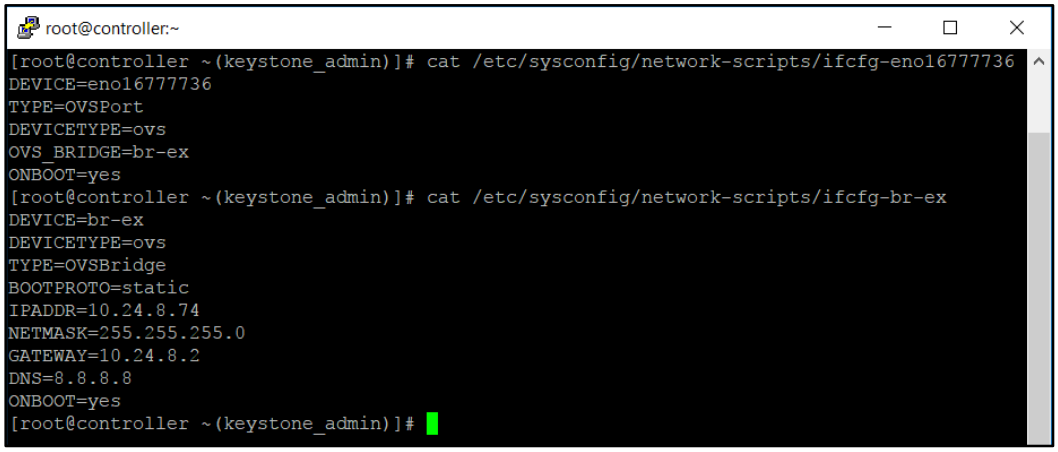


Figura 66. Interfaces habilitadas para Neutron – OVS

Una vez que las interfaces se encuentran correctamente configuradas y habilitadas para trabajar con el módulo de red de OpenStack, es posible realizar la verificación de los agentes de red que intervienen en la virtualización de los servicios de red en el nodo de control y en los nodos esclavos destinados a usarse como recursos de computo. La Figura 67 detalla el estado de los agentes de red y otros componentes necesarios para el funcionamiento de la red virtual.

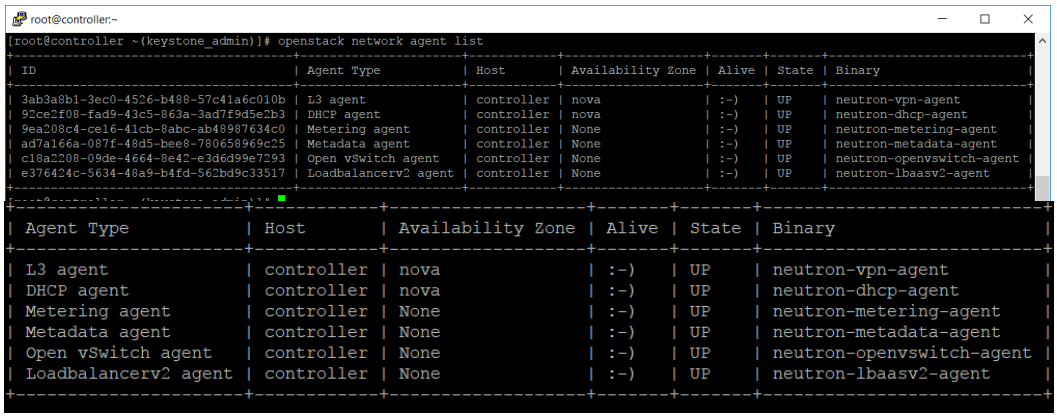


Figura 67. Agentes del módulo de red Neutron.

En este punto es necesario dirigirse al dashboard de OpenStack para crear una red de acceso externo y un router virtual como la indicada en la Figura 68. Esta permitirá a las instancias alcanzar los destinos fuera de la red local, es decir internet. Parte de esta configuración conlleva la correcta vinculación del bridge br-ex con la interfaz de acceso física. Si esta configuración es errónea bloqueara todas las conexiones desde y hacia las instancias que se encuentran dentro de la plataforma de virtualización. Una forma de identificar si existen errores es la utilización de la línea de comandos del servidor para tener acceso a los dispositivos de red virtuales y realizar pruebas de echo hacia adentro y fuera de la red virtual. Después de que se realice la configuración de la red externa y se haya creado el enrutador virtual puede ser necesario volver a recrear la instancia y reiniciar el sistema ya que los dispositivos virtuales no se crean realmente sino hasta que una instancia cree un nuevo puerto y se vincule a este. El proceso completo de creación y administración de redes virtuales se detalla en el ANEXO C.

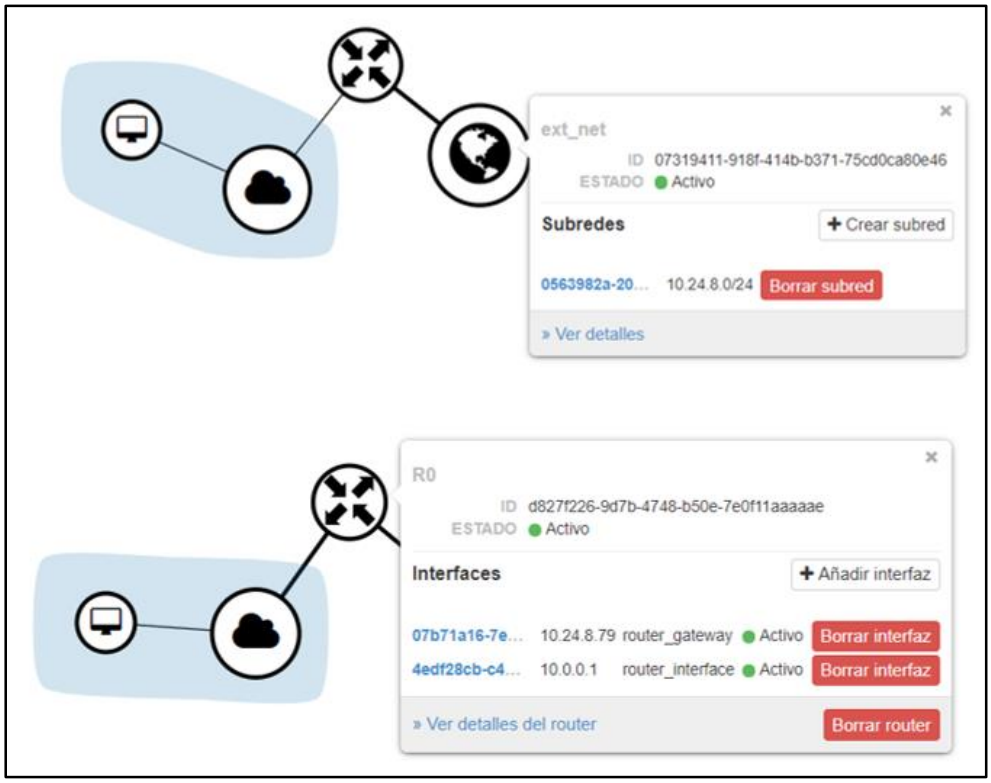


Figura 68. Router virtual conectado a la red física y a la red virtual de OpenStack.

Una de las tareas más importantes una vez que se ha configurado el acceso de las instancias a la red externa es la elaboración de las políticas de acceso y seguridad mediante el gestor de grupos de seguridad que proporciona el componente Neutron. Para la creación de estas reglas es necesario dirigirse a la sección redes del proyecto administrador y en la pestaña Grupos de Seguridad encontraremos el grupo por defecto en donde se debe eliminar todas las reglas y agregar únicamente las políticas de acceso para la red de administración y la red interna de la institución.

Inicialmente el grupo de seguridad se encontrará en blanco y el acceso externo a las instancias estará completamente restringido. La primera regla consiste en permitir el acceso a través de SSH desde las redes de confianza, la segunda regla permite el tráfico ICMP en las redes de confianza. Finalmente se asigna una dirección IP flotante a cada instancia que necesite administración remota,

de esta manera el servicio quedara abierto a través del mismo puerto, pero en la IP flotante generada a partir del pool de la red física. Las reglas del grupo de seguridad se pueden apreciar en la Figura 69.

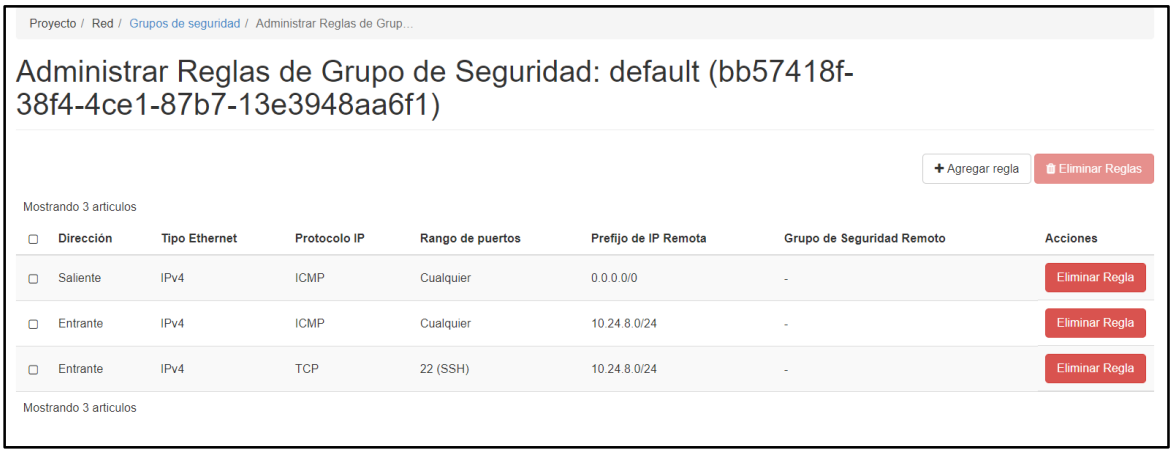


Figura 69. Reglas básicas en el Grupo de Seguridad de la red virtual.

El paso final de esta etapa es verificar que en el controlador de la plataforma cloud se encuentre funcionando correctamente el bridge de red. Es necesario que se encuentre en perfecto funcionamiento ya que posteriormente se convertirá en la interfaz principal de comunicación con los switch virtuales haciendo las veces de servidor de virtualización de red.

La verificación se realizará mediante la herramienta Open vSwitch Configuration Tool. Primero ejecutaremos el comando *show* para comprobar que todos los elementos hasta aquí virtualizados se encuentren en perfecto estado y correctamente configurados, el resultado de la ejecución de este comando se muestra en la Figura 70. Luego procederemos a revisar uno a uno los componentes de bridge externo asegurándonos que no existan entradas de puerto incompletas. Por ultimo revisaremos que no existan errores en la tabla de flujo del bridge usando la utilidad ovs-ofctl.

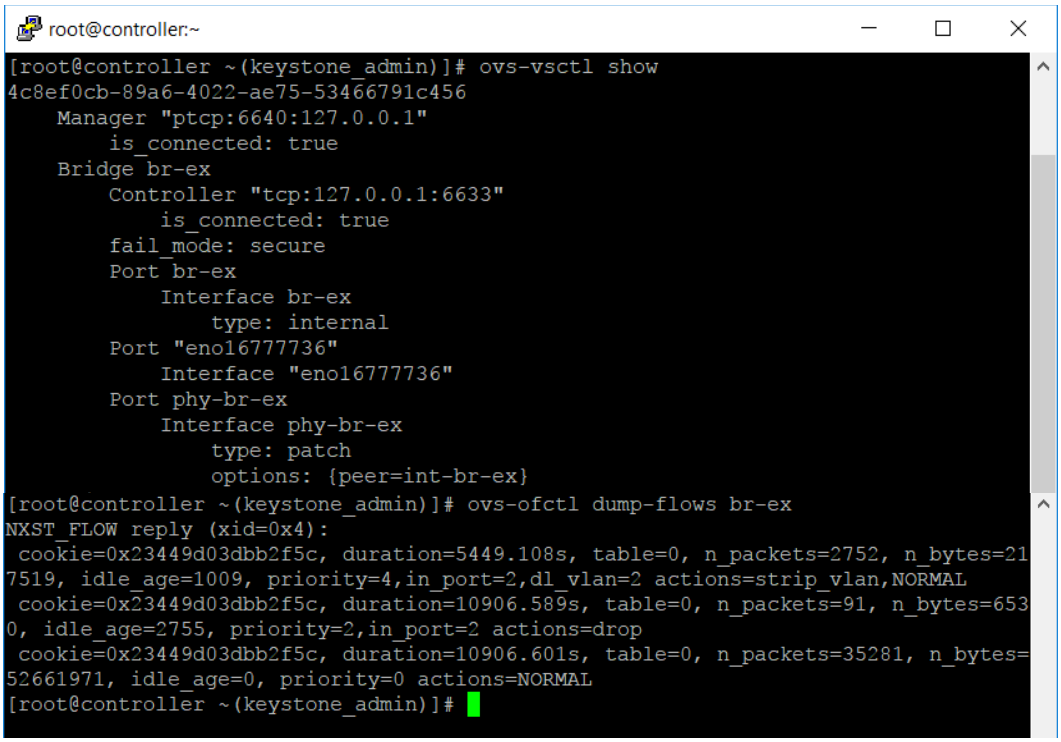


Figura 70. Resultado del comando ovs-vsctl y ovs-ofctl.

3.3. EVALUACIÓN DE CONTROLADORES PARA SDN

Existen múltiples soluciones para controladores de redes SDN tanto de código abierto como de pago. Uno de los objetivos del presente proyecto es brindar una solución libre, mediante el uso de soluciones basadas en software y que se adapten al hardware que se encuentra disponible actualmente. Por esta razón se dejará de lado las soluciones empresariales de Cisco, HP y VMware que solo poseen paquetes disponibles para hardware del fabricante.

Para la evaluación de los controladores basados en software se usa un ambiente experimental a base de máquinas virtuales sobre el hipervisor VMware Workstation. En esta sección se analizan las funcionalidades de cada controlador usando una herramienta muy útil para simulación de redes llamada mininet. Se descarga una imagen prediseñada para enfocarme directamente en las características funcionales del software de control. Se usan dos máquinas virtuales. En la primera

se instalan uno por uno los controladores para las respectivas pruebas, y en la segunda se levanta el simulador de redes Mininet que se encargara de proporcionar los dispositivos de red que aparecerán en el controlador una vez se hayan integrado ambos servicios.

En la sección 2.4 se listan los controladores OpenFlow y SDN más conocidos. No todos los controladores de la lista son candidatos para ser evaluados, ya que varios se encuentran descontinuados o no entran en el grupo de basados en software. Los controladores que se someterán a evaluación para la selección del controlador que se implementara en la red SDN serán:

- RYU
- FLOOFLIGHT
- OPENDAYLIGHT
- ONOS

3.3.1. EMULADOR MININET

Mininet es un emulador de redes, que ejecuta una colección de módulos para la creación de hosts, conmutadores, enrutadores y enlaces de red sobre el kernel de Linux. Utiliza virtualización liviana para hacer que un solo sistema se vea como una red completa, replicando el mismo kernel, sistema y código de usuario.

Un host mininet es capaz de comportarse como una maquina real, se puede entrar en el mediante SSH y ejecutar programas arbitrarios incluidos los que se encuentran en el sistema subyacente. Los programas que ejecuta pueden enviar paquetes a través de un elemento que simula el comportamiento de una interfaz ethernet real. Los paquetes se procesan mediante un conmutador virtual, generalmente Open vSwitch, y permite el uso de una gran variedad de protocolos comunes

en las redes de datos incluyendo OpenFlow. El proceso de instalación de la herramienta se detalla en el ANEXO D. (Mininet Project, 2017)

3.3.2. PRUEBA CON RYU

La evaluación del controlador RYU se realiza mediante su instalación en una máquina virtual con el sistema operativo Ubuntu 16.04.3 LTS, sobre el hipervisor VMware Workstation. El proceso inicial consiste en obtener los binarios del controlador desde los repositorios de la distribución. Se debe tomar en cuenta que no todas las distribuciones tienen soporte para las dependencias del software.

Una vez que se ha descargado el software de RYU es necesario instalar ciertas dependencias extra que no están expuestas en la documentación oficial. Esto sucede a causa del poco soporte que reciben estos paquetes debido a su usabilidad en ambientes de producción. Cuando se han satisfecho todas las dependencias se ejecuta el software de controlador desde la línea de comandos, y si no existe ningún error el resultado será similar a la imagen de la Figura 71. El proceso de instalación se encuentra expuesto con mayor claridad en el ANEXO D.

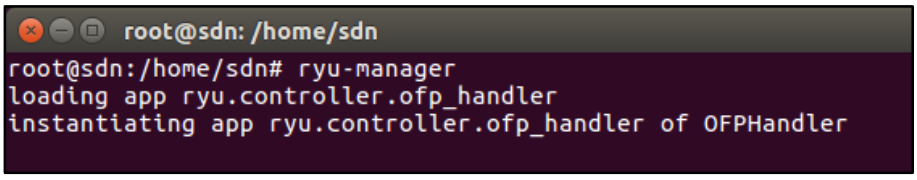


Figura 71. Handler de RYU en estado funcional.
Fuente: Elaborado por el usuario.

Con el controlador en estado funcional, se permite usar los módulos que se ofrecen de manera separada. Para la realización de la prueba se usa el módulo de conmutación simple junto a una red simulada a partir del software de simulación mininet alojado en otro equipo. Primero ejecutaremos

la instrucción para inicializar el controlador, y luego arrancamos la simulación para poder apreciar los mensajes de notificación instantánea que genera RYU. La Figura 72 ilustra el estado de comunicación del controlador con los switches virtuales que se simulan en mininet.

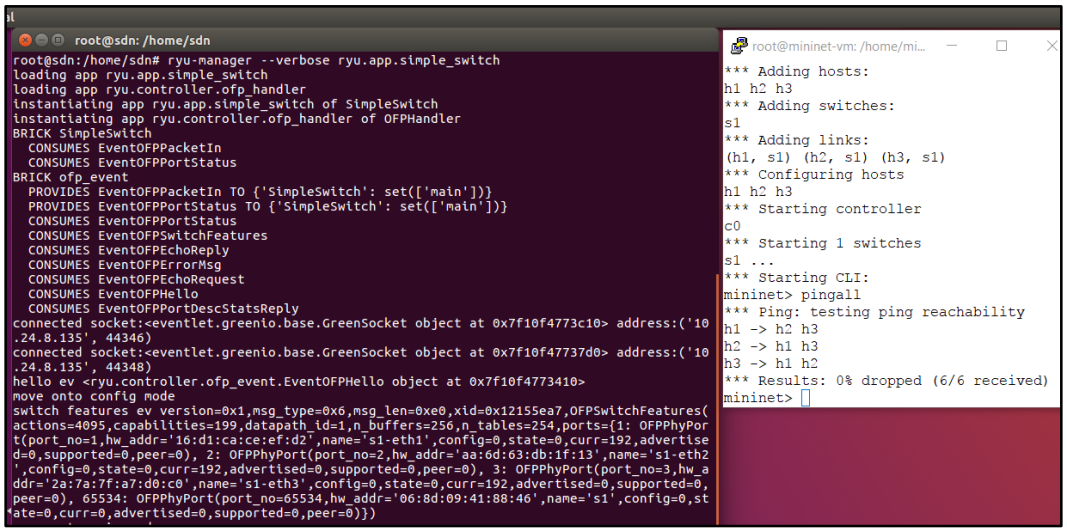


Figura 72. Comunicación establecida entre RYU y mininet.

La segunda parte de la prueba consiste en realizar una breve captura de paquetes para comprobar cómo se lleva a cabo la sesión OpenFlow y que tipo de paquetes se están transmitiendo. Para ello se emplea la herramienta wireshark, instalada en el mismo servidor en el que se aloja el software de control. Mientras el controlador y la red se encuentran activos se ejecuta una nueva captura de paquetes. Inmediatamente se debe apreciar una serie de mensajes etiquetados como OpenFlow en la columna protocolo como se muestra en la Figura 73. En caso de existir mayor número de mensajes de otros protocolos, se aplica un filtro mediante la selección de uno de los paquetes capturados usando la opción seguir>conversación TCP. Así es posible revisar la secuencia de la sesión OpenFlow rápidamente. La versión de la especificación que usa RYU en este caso es la 1.0.

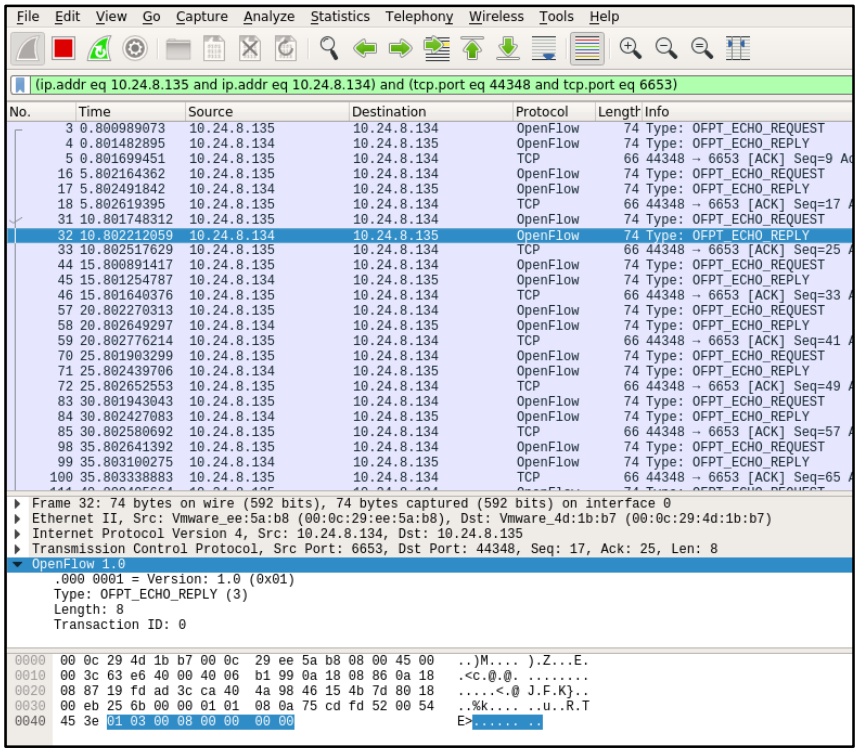


Figura 73. Captura de tráfico OpenFlow de la comunicación entre RYU y mininet.
Fuente: Wireshark Packet Capture.

3.3.3. PRUEBA CON FLOODLIGHT

Para la evaluación de características y funcionalidades del controlador Floodlight se usa una máquina virtual sobre el hipervisor VMware Workstation, en la que se ha instalado el sistema operativo Ubuntu 16.04.3 LTS. Para iniciar es necesario descargar el set de herramientas de desarrollo de Linux y las utilidades git y ant. Esto con el objetivo de compilar el código fuente del proyecto Floodlight. Estos pasos se encuentran especificados en el ANEXO D.

Lo siguiente es ejecutar la aplicación escrita en java para que el controlador inicie los módulos y servicios internos. En cuanto el controlador esté listo es posible ingresar a la interfaz web de la Figura 74, en donde se encuentran los apartados de administración y configuración, así como los dispositivos de red y las tablas de flujo.

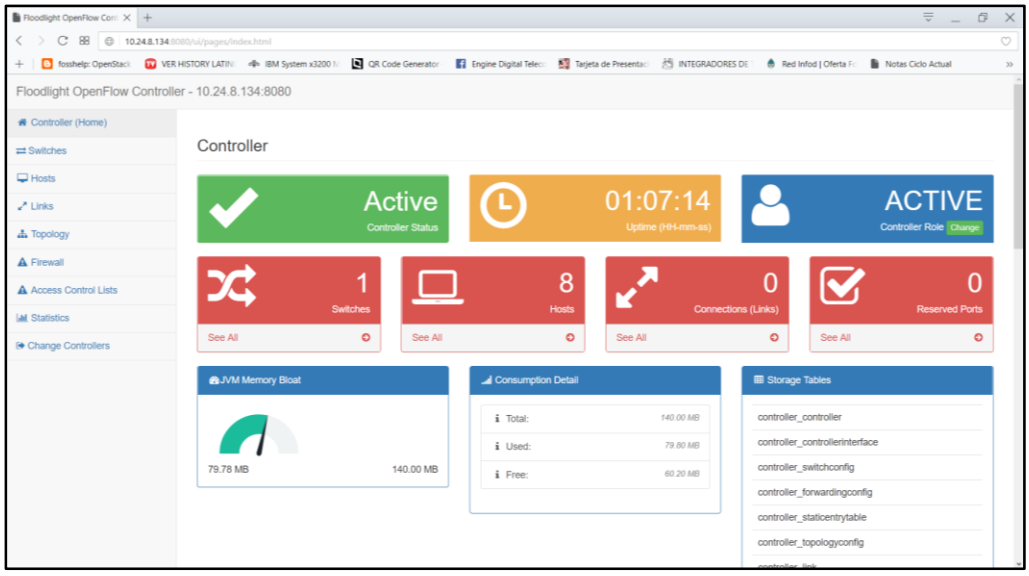


Figura 74. Interfaz de usuario de Floodlight.
Fuente: Floodlight Openflow Controller.

Una prueba básica se puede realizar usando el simulador de redes mininet. Este simulador trabaja a base de scripts de ejecución o comandos parametrizados. En este caso se ingresa una instrucción construida con los parámetros básicos, como la dirección IP del controlador, el puerto de comunicación, el tipo de switch y la versión de OpenFlow como se aprecia en la Figura 75.

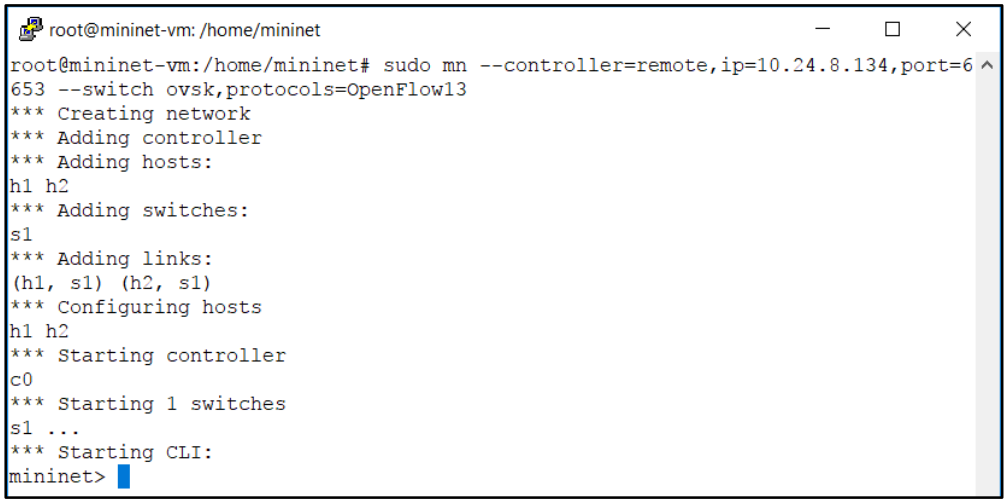


Figura 75. Instrucción para crear una red OpenFlow de prueba.

El parámetro más importante de la instrucción ejecutada para la simulación de la red es *controller*, ya que este es el que permite definir cuál es la dirección del servidor donde se encuentra el software de control. Si no se encontraron errores el controlador será capaz de establecer la comunicación con el switch que se ha simulado y permitirá visualizar sus características, y a partir de esta información será capaz de generar la topología grafica de la red. En la Figura 76 se ejecuta una prueba de ping entre los dos hosts que se crean automáticamente en cada puerto del switch virtual, e inmediatamente la interfaz gráfica de usuario muestra las estadísticas del switch y las entradas en las tablas de flujo.

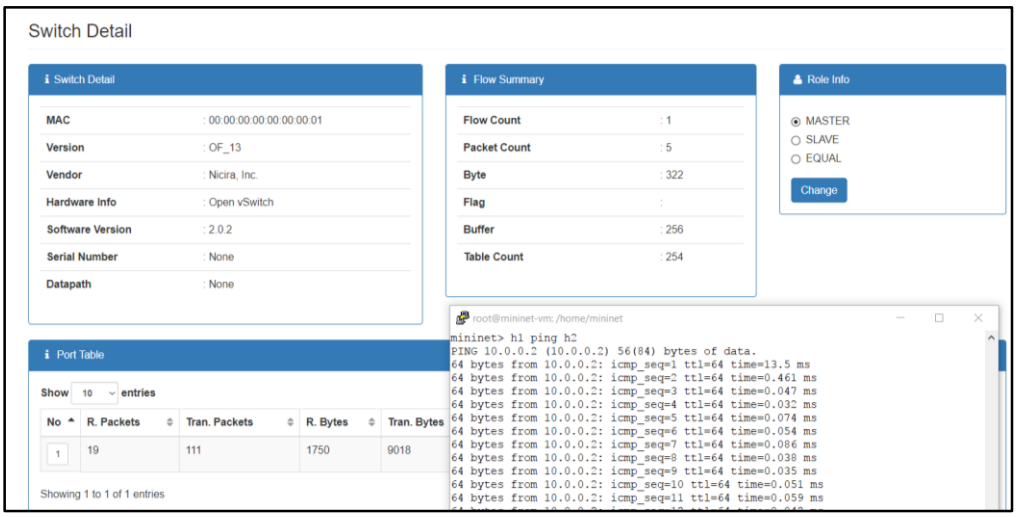


Figura 76. Estadísticas del switch de prueba en Floodlight.
Fuente: Floodlight Openflow Controller.

La segunda prueba es técnica, y se realizara mediante el uso de la herramienta de captura de paquetes wireshark. En realidad, no es necesario contar con un entorno grafico en el servidor que hace de controlador ya que también existen herramientas de captura y análisis de paquetes en versión de línea de comandos. Para efectos experimentales se ha usado la distribución de Ubuntu en su versión gráfica.

Una vez que se ha instalado el paquete wireshark en el servidor, se procede a ejecutar una nueva captura. Normalmente Wireshark inicia la captura general del tráfico en la interfaz seleccionada, pero en este caso se usa un filtro que muestra únicamente los paquetes pertenecientes a la sesión OpenFlow. Esto se logra buscando al menos un paquete etiquetado como Openflow en la columna protocolo y seleccionando la opción Seguir Conversación >TCP. Hecho esto se muestra un resultado como el de la Figura 77. Wireshark presentará únicamente los mensajes de la sesión OpenFlow, en este caso de la versión 1.3.

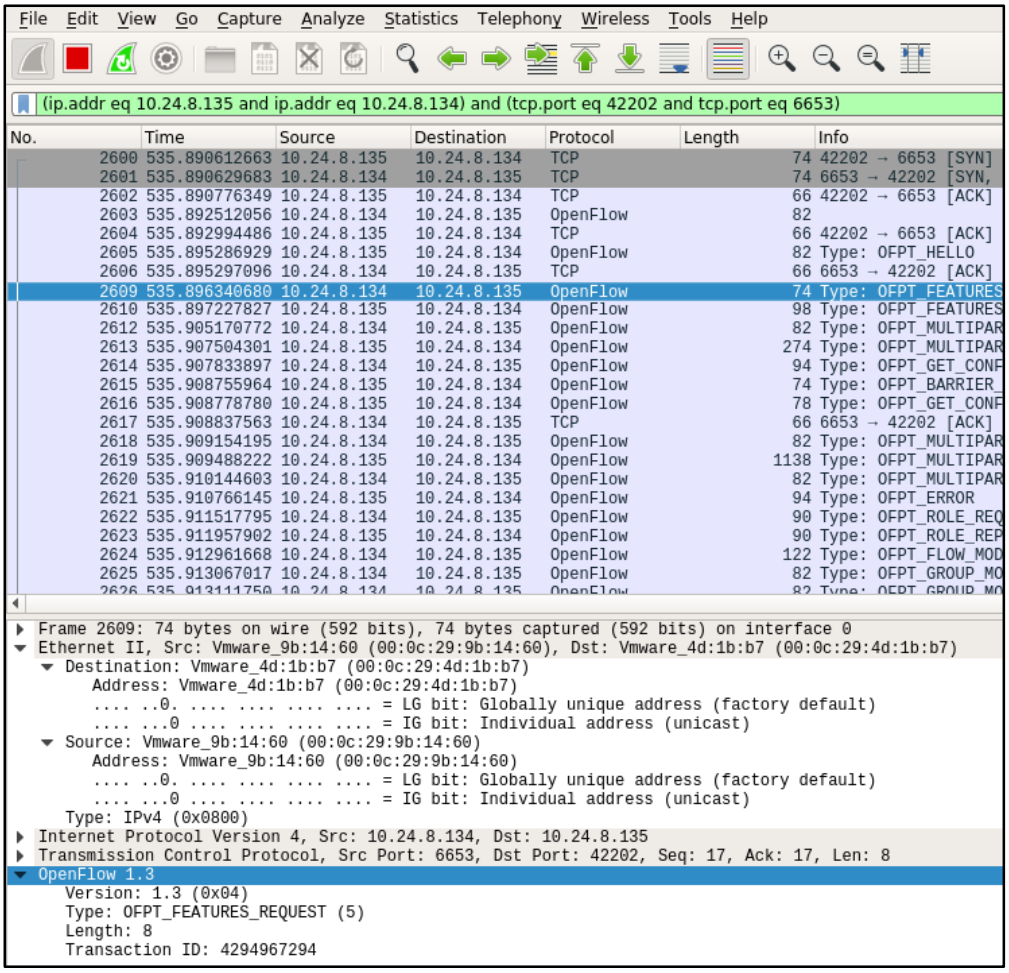


Figura 77. Captura de trafico sobre la interfaz del controlador Floodlight.
Fuente: Wireshark Packet Capturer.

3.3.4. PRUEBA CON OPENDAYLIGHT

Durante el proceso de prueba del controlador OpenDaylight se utilizan dos máquinas virtuales. La primera hospeda exclusivamente el software de control de OpenDaylight y la segunda contiene el simulador de redes mininet. Se crea un ambiente sencillo específicamente para efectos empíricos a base de elementos virtuales con características suficientes para suplir los requisitos de las pruebas. Ambas máquinas virtuales funcionan con el sistema operativo Ubuntu 16.04.3 LTS.

OpenDaylight está escrito en java por lo que requiere ciertos componentes específicos para funcionar, entre estos está el kit de desarrollo de Oracle Java. Antes de iniciar debemos asegurarnos que todas las dependencias están satisfechas y no existe ningún error de compilación en el programa. Una vez hecho esto arrancamos el software e ingresamos a la consola de programación del controlador. Los pasos de instalación y acondicionamiento del software se explican en el ANEXO D.

Inicialmente OpenDaylight no tiene ningún módulo instalado ya que este cuenta con una colección amplia de componentes. Por esta razón procedemos a agregar los elementos básicos para la creación del ambiente de prueba. Con los módulos ya cargados el controlador adquiere nuevas capacidades entre ellas la creación de entradas de flujo dinámicas, creación de tablas de flujo automáticas y generación de la topología gráfica. El componente más evidente es *dlux*, que es el encargado de publicar la interfaz gráfica del controlador, ilustrada en la Figura 78.

A partir de una instalación limpia del controlador OpenDaylight en la primera máquina virtual, se efectúa la adición de los componentes básicos para su correcta operación. Entre las características instaladas se encuentran las funciones de capa dos, y el soporte para las versiones más estables de OpenFlow. Para comprobar que el controlador se encuentra funcional se crea una

topología en el simulador mininet y se establece el parámetro controlador con la IP de la primera máquina virtual, donde inmediatamente se sincroniza el estado de la red y se genera el gráfico de la topología.

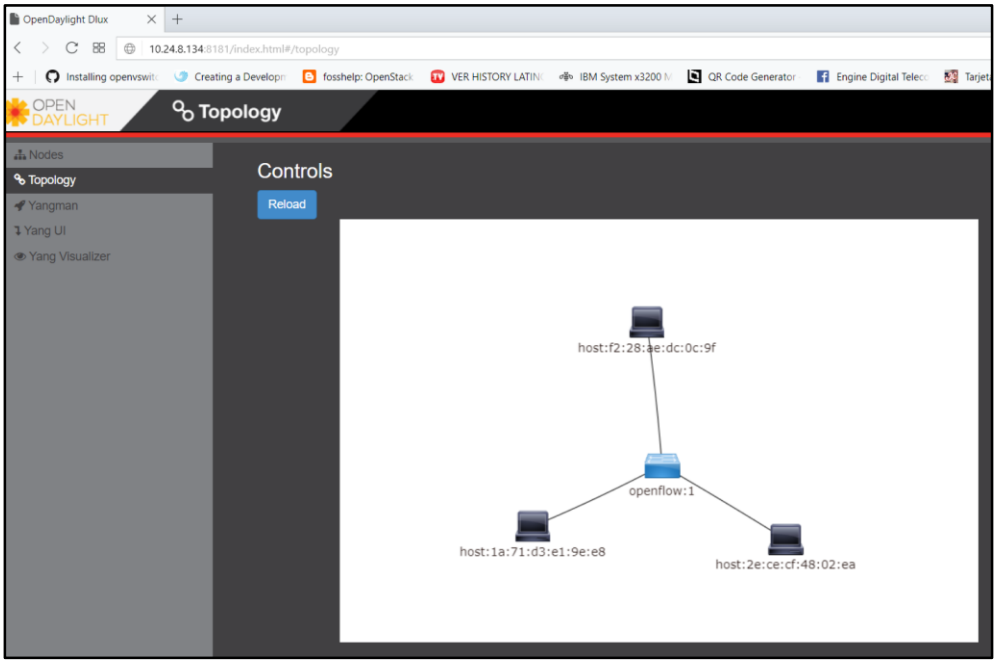


Figura 78. Interfaz DLUX de OpenDaylight.
Fuente: OpenDaylight SDN Controller.

Los componentes de OpenDaylight son de una constitución muy robusta ya que son depurados constantemente. DLUX cuenta con un nivel de personalización bastante alto, permitiendo la creación de nuevas API's y componentes usando el lenguaje JSON. La mayoría de los módulos son operacionales y muy pocos son de configuración.

YANG es el lenguaje legible por humanos para describir dispositivos de datos y modelos de servicio. OpenDaylight genera dinámicamente REST API's a partir de modelos YANG proporcionando a los desarrolladores de aplicaciones conjuntos API de referencia. YANGMAN

es una herramienta incorporada en las versiones más recientes de OpenDaylight. Ofrece formularios de interfaz de usuario generados dinámicamente y representaciones JSON nativas basadas en RESTCONF como se observa en la Figura 79, que permiten la configuración de los parámetros programables del controlador a través de la interfaz gráfica. (CiscoDevNet, 2017)

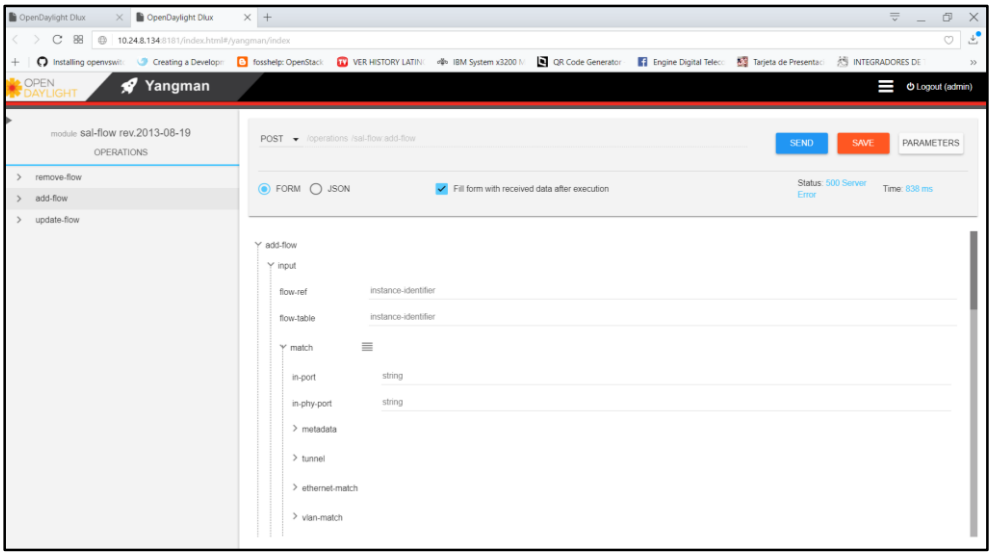


Figura 79. Herramienta YANGMAN integrada a OpenDaylight.
Fuente: OpenDaylight SDN Controller.

Una vez que se ha verificado que el controlador y sus módulos se encuentran activos, se procede a generar tráfico de prueba entre las interfaces de los dispositivos de red simulados y el controlador OpenDaylight. Instantáneamente las tablas de flujo se actualizan y muestran la cantidad de paquetes transmitidos. Para visualizar más profundamente el estado de la sesión OpenFlow se usa el analizador de paquetes wireshark en la interfaz física del controlador, y se filtra los mensajes para observar únicamente los pertenecientes a la conversación OpenFlow. En la ventana de la Figura 80 se pueden apreciar los parámetros más importantes de la sesión OpenFlow como la versión de la especificación, los mensajes de la sesión y el intercambio de mensajes de control que indican que el modo de funcionamiento es *OpenFlow over TCP*.

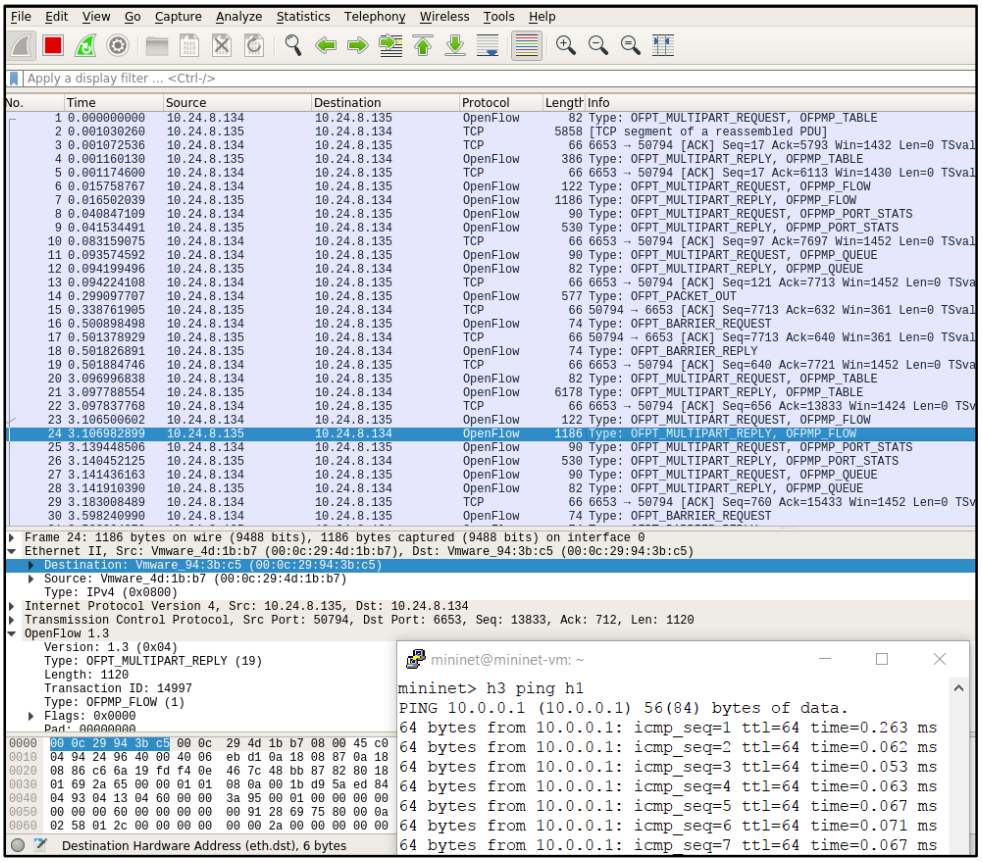


Figura 80. Sesión OpenFlow de OpenDaylight en Wireshark.
Fuente: Wireshark Packet Capturer.

3.3.5. PRUEBA CON ONOS

Para las pruebas de funcionalidad de Open Network Operating System adecuaremos un ambiente con dos máquinas virtuales en las que instalaremos el software del controlador y la herramienta mininet. ONOS es una plataforma escrita en java y usa OSGi para la administración de sus funciones al igual que OpenDaylight. Sus características se cargan usando su tiempo de ejecución denominado Karaf.

La plataforma es modular y está construida con aplicaciones particulares ancladas a un componente graficador de topologías de nivel superior. Antes de empezar a utilizar el controlador, es necesario habilitar las características relacionadas con el reenvió inteligente de paquetes, así

como el soporte para OpenFlow. Este proceso esta detallado en el ANEXO D en la sección correspondiente a ONOS.

Una vez que los módulos básicos se encuentran activados, desde la máquina virtual con mininet se genera una topología de prueba conectando uno o más conmutadores al controlador apuntando el simulador a la IP de ONOS. Una vez que ejecutamos la instrucción de simulación en mininet, desde la consola de ONOS comprobamos que el controlador aprendió nueva información de los hosts conectados, el resultado de esta consulta se muestra en la Figura 81.

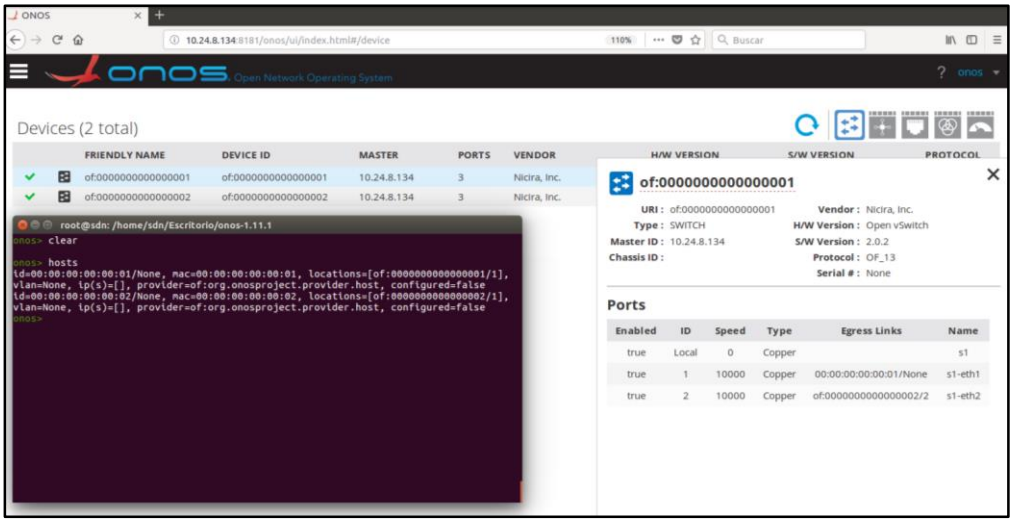


Figura 81. Descubrimiento de hosts en ONOS Controller.
Fuente: Open Networking Operating System.

ONOS cuenta con una biblioteca de nódulos que pueden ser activados tanto desde la línea de comandos como desde la interfaz gráfica. Muchos de estos complementos tienen funciones desarrolladas para escenarios específicos, por lo que no es necesario habilitar todos al mismo tiempo. Para aprender los enlaces en la topología y calcular las rutas de los hosts, ONOS utiliza el protocolo LLDP, y por el momento onos-app-fwd es el módulo encargado de programar los flujos de manera reactiva. Si iniciamos una solicitud de ping entre dos hosts y detenemos el este servicio el ping fallará como en la imagen de la Figura 82.

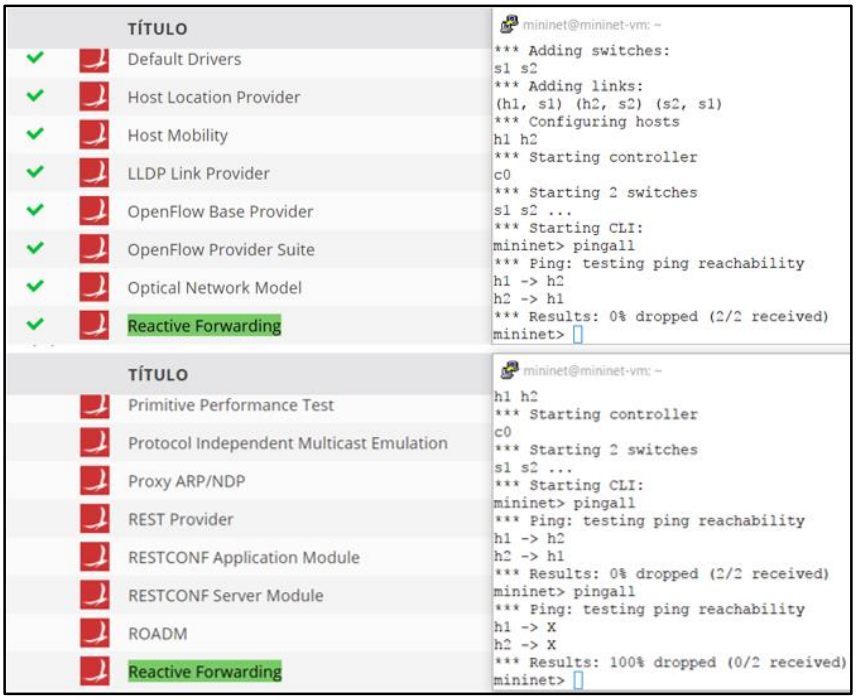


Figura 82. Solicitud de ping con el módulo de reenvío de ONOS activo e inactivo.
Fuente: Open Networking Operating System.

Una vez comprobada la funcionalidad del controlador de manera exitosa, se procede a realizar el análisis de los paquetes inmersos en la sesión OpenFlow entre ONOS y los elementos de red simulados en mininet. Este análisis se desarrolla mediante el software wireshark, en el que usamos la interfaz de red física del servidor que hospeda al controlador de red para capturar el tráfico generado en la comunicación mediante OpenFlow. El resultado obtenido son una serie de mensajes etiquetados como OpenFlow en la columna protocolo que contienen los campos de las cabeceras y la información propia de la especificación como indica la Figura 83.

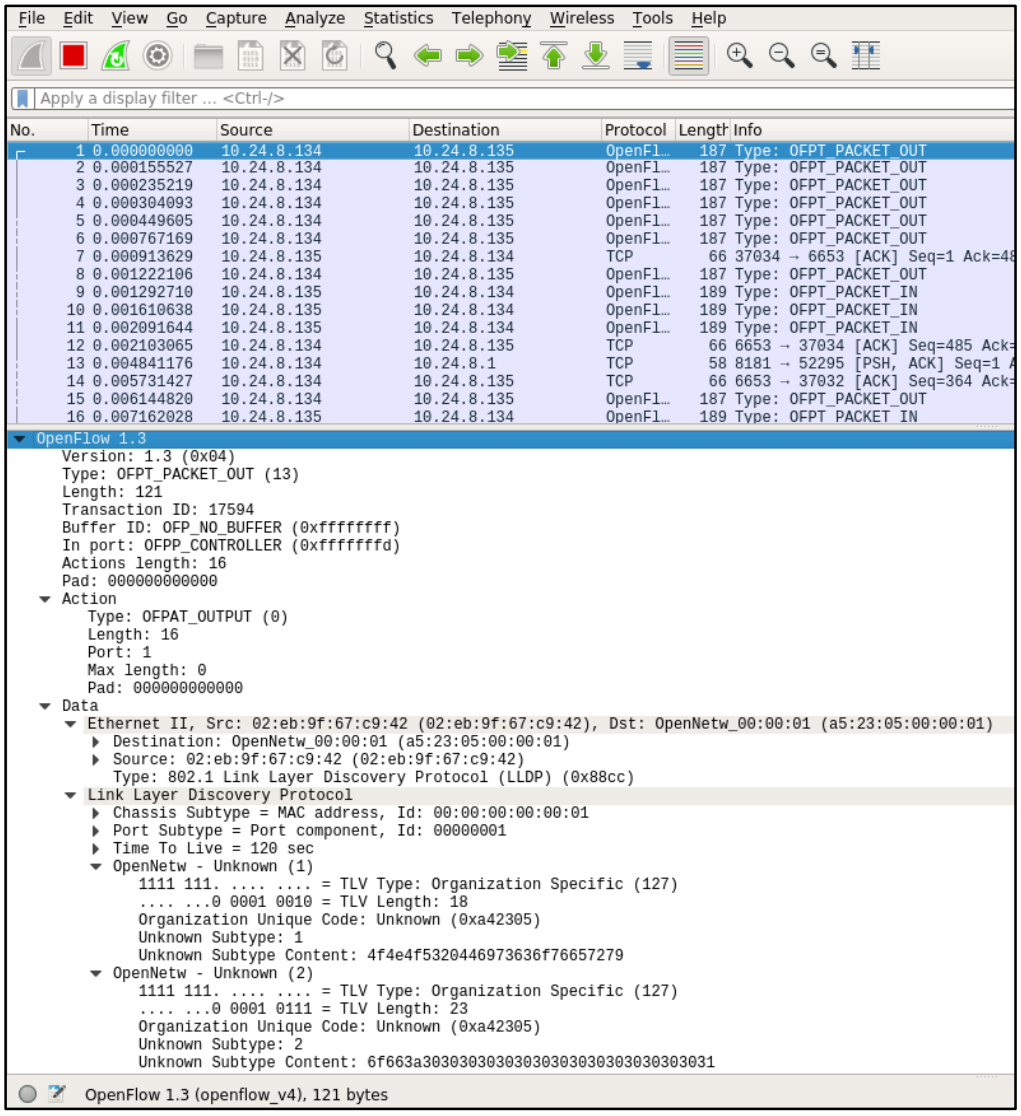


Figura 83. Mensaje PACKET_OUT generado por ONOS hacia el conmutador simulado.
Fuente: Open Networking Operating System.

3.4. SELECCIÓN DEL SOFTWARE CONTROLADOR

La selección del software controlador que se usara para el despliegue no se realiza de manera arbitraria. Para esta tarea es necesario usar un método de selección que permita escoger la mejor opción de acuerdo a los requerimientos del proyecto. En este caso se usará un modelo de selección en base a los parámetros impuestos por la norma ISO/IEC/IEEE 29148 en su enmienda del año 2011. Esta norma presenta los requisitos y parámetros para la selección de software para un

proyecto de implementación. Dichos parámetros se especifican en la sección “9.5 Especificación de los Requisitos de Software (SRS)”. Esta se encuentra adjunta en el ANEXO A-1.

3.4.1. PARAMETROS DE SELECCIÓN DEL SOFTWARE CONTROLADOR

Generalmente el proceso de selección de un software obedece a las necesidades de establecer criterios de selección dentro de las actividades previas al desarrollo. Estos criterios comprenden un conjunto de parámetros estrechamente ligados a las necesidades de los usuarios, y sirven de referencia para la evaluación y, por lo tanto, son la clave para la decisión de selección.

3.4.1.1. Propósito

El propósito principal del software es mantener el control de red definida por software, permitiendo que la administración resulte mucho más sencilla, a la vez que proporcione un método de verificación de la actividad en los elementos participantes y la visualización de las conexiones establecidas sobre la infraestructura de red.

3.4.1.2. Alcance

El software debe ser capaz de administrar una red definida por software, por medio el uso de reglas de tráfico de datos y reglas de control de acceso. Debe también ser capaz de garantizar el funcionamiento correcto de una red SDN a base de la monitorización y gestión centralizada desde un solo punto de la red.

3.4.1.3. Perspectiva del Producto

Se proyecta implementar un sistema centralizado que permita controlar y distribuir reglas de tráfico a los dispositivos que lo soporten. El sistema de control de red a desplegarse es un software

independiente, pero se tiene programada la integración con un sistema de cloud privado previamente instalado, con soporte para redes definidas por software.

3.4.1.4. Funciones del Producto

EL controlador debe cumplir con las siguientes funciones:

- Permitir el tráfico de datos de forma manual, según la demanda de los usuarios.
- Permitir la conexión directa con switch virtuales por medio de un puerto específico.
- Descubrir automáticamente los dispositivos SDN y sus respectivas características.
- Mapear las conexiones activas e inactivas de un equipo que forma parte de la SDN.
- Permitir el desarrollo de aplicaciones de integración de funciones para la SDN.

3.4.1.5. Características de los usuarios

El software debe presentar características pertenecientes a capas de nivel superior, esto debido a que el grupo de usuarios principales de la red será el personal de administración de redes. Este grupo de usuarios tiene experiencia en el manejo de software, pero no de la infraestructura SDN en su totalidad.

3.4.1.6. Limitaciones

Las limitaciones del software controlador SDN están dadas mayormente por variables no controladas, y son:

- Se cuenta únicamente con un servidor disponible para el desarrollo de la porción de control de la SDN, por lo que no se pueden hacer pruebas paralelas.

- La cantidad de recursos utilizables del servidor es muy baja.

- El acceso al servidor se debe hacer mayormente de manera local.

3.4.1.7. Suposiciones y dependencias

Los requisitos descritos en este documento pueden cambiar, ya que el proceso es dinámico. Durante el despliegue de la SDN se supone que:

- El controlador es compatible con el sistema operativo Linux, ya que el proyecto se desarrolla en software libre.

- El servidor donde se aloja el controlador de red siempre tiene conexión a internet.

- La dirección IP asignada al servidor controlador es privada y parte del pool de la DMZ de la Universidad Técnica del Norte.

- El controlador tiene soporte para la integración con NFV y plataformas cloud.

3.4.1.8. Requisitos de las interfaces

Los requisitos de las interfaces están relacionados directamente con el hardware. A continuación, se listan estos requisitos de acuerdo a los parámetros afines.

- **Interfaces de Usuario:** el software debe poseer interfaz gráfica, y permitir la visualización de la información de la red.

- **Interfaces de hardware:** el servidor debe contar con una NIC de red, que permita establecer la conexión a internet y a la red de infraestructura local de la Universidad.

- **Interfaces de software:** el software debe permitir la creación de interfaces virtuales y el modo de comunicación de la red SDN, en este caso la especificación OpenFlow.

- **Restricción de Memoria:** el software debe consumir una mínima cantidad de recursos, para garantizar el funcionamiento en caso de condiciones de estrés o saturación de red.

3.4.1.9. Requisitos funcionales

La Tabla 10 describe los requisitos funcionales para la selección del software controlador de la red definida por software. La prioridad está dada por valores del 1 al 3, en donde 3 es el valor máximo y 1 es lo mínimo. Un valor de cero significa que no existe soporte para la característica. En el casillero final se calcula el valor total que determina que controlador es el apropiado.

Tabla 10.

Requerimientos Funcionales del controlador

Nro.	Nombre	Característica	Descripción	Prioridad			
				RYU	**Floodlight**	**OpenDaylight**	**ONOS**
REQ 01	Openflow	Soporte para Openflow 1.0 - 1.3	El controlador debe soportar las comunicaciones a través de OpenFlow.	2	3	3	3
REQ 02	Plataforma	Soporte para OpenStack	El controlador debe soportar la integración con entornos cloud.	0	2	3	2
REQ 03	Virtualización	Soporte para Open vSwitch	El controlador debe soportar el uso de switches basados en el Software OVS.	2	3	3	3
REQ 04	Sistema Operativo	Soporte para varios sistemas operativos	El controlador debe soportar su uso bajo software libre y licenciado.	1	2	3	3
REQ 05	API REST	Arquitectura de desarrollo web	El controlador debe poseer una API REST, para la manipulación de la red.	0	2	3	3
REQ 06	Bucles	Repeticiones de Paquetes	El controlador debe tener mecanismos de control de bucles de datos	0	2	3	3
REQ 07	Multiproceso	Ejecución de uno o varios procesos	El procesador debe tener la capacidad de ejecutar procesos paralelos.	0	2	3	3
		TOTAL		5	16	21	20

3.4.1.10. Requisitos No Funcionales

La Tabla 11 describe los requisitos no funcionales que el controlador debe cumplir, para ser elegible. El puntaje de esta tabla influye en gran medida en la decisión de selección.

Tabla 11.

Requisitos no funcionales del controlador.

Nro.	Nombre	Característica	Descripción	Prioridad			
				RYU	**Floodlight**	**OpenDaylight**	**ONOS**
REQ 08	Java	Lenguaje de programación	EL controlador debe estar desarrollado en java.	0	3	3	3
REQ 09	Licencia	Licencia GPL	El controlador debe manejar licencias de código abierto.	1	3	3	3
REQ 10	GUI	Interfaz Gráfica para el usuario.	El controlador debe poseer una GUI intuitiva y fácil de usar.	0	2	3	3
REQ 11	Instalación	Persistencia en equipos	El controlador debe ser fácil de instalar.	1	2	3	3
REQ 12	Simplicidad	Facilidad de uso.	El controlador debe ser de fácil manipulación.	1	2	3	2
REQ 13	Comunidad Activa	Personas que modifican y actualizan	El controlador debe poseer documentación actualizada y con soporte de la comunidad.	0	2	3	2
REQ 14	Mercado	Tiempo de actividad	El controlador debe contar con actualizaciones periódicas.	2	2	3	3
		TOTAL		5	16	21	19

3.4.1.11. Requisitos de Rendimiento

El software del controlador se instalará directamente sobre el sistema operativo del servidor designado para esta tarea. Al momento de monitorear la red definida por software y los dispositivos que forman parte de ella, el trafico excesivo de datos no debe afectar el funcionamiento normal del equipo.

3.4.2. JUSTIFICACIÓN DE LA SELECCIÓN

De acuerdo a los resultados de las natrices de evaluación para la selección de software, el que cumple con todos los requisitos establecidos por la norma ISO/IEC/IEEE 29148 es OpenDaylight. Este presenta las mejores funciones y características, por lo que se adapta perfectamente al despliegue planteado en este proyecto.

OpenDaylight presenta la ventaja de ser compatible con switches físicos y virtuales, presenta una interfaz web amigable y posee el soporte para API REST. Otra característica importante es su compatibilidad con más de un sistema operativo y las versiones más actuales de java. Su instalación es sencilla, su desarrollo es muy activo y además cumple con los requisitos funcionales y no funcionales completamente. Esto lo vuelve un software muy robusto lo que garantiza un desempeño optimo y eficiente cubriendo las necesidades de la infraestructura de la SDN.

3.4.2.1. Parámetros técnicos de OpenDaylight

EL controlador de red OpenDaylight se eligió debido a que cumple tres parámetros fundamentales:

- **Adaptabilidad:** al estar escrito en java permite el desarrollo de aplicaciones sobre cualquier plataforma que lo soporte.

- **Funcionalidad:** tiene la capacidad de administrar switch físicos y virtuales, a la vez que maneja flujos de tipo reactivo y proactivo.

- **Usabilidad:** posee documentación extensa acerca de su desarrollo, instalación y configuración, además del apoyo de la comunidad.

3.5. IMPLEMENTACIÓN DEL CONTROLADOR SDN

Esta sección está destinada a detallar el proceso de implementación del software de control de redes definidas por software denominado OpenDaylight, el cual ha sido seleccionado mediante los parámetros establecidos por la norma ISO/IEC/IEEE 29148. Entre los pasos del proceso que se describen se encuentra el estudio del hardware que se utiliza como base, así como los componentes o agentes externos necesarios para establecer la comunicación mediante la especificación OpenFlow entre el controlador y los dispositivos de conmutación de la SDN. La Figura 84 muestra un esquema de la implementación del controlador.

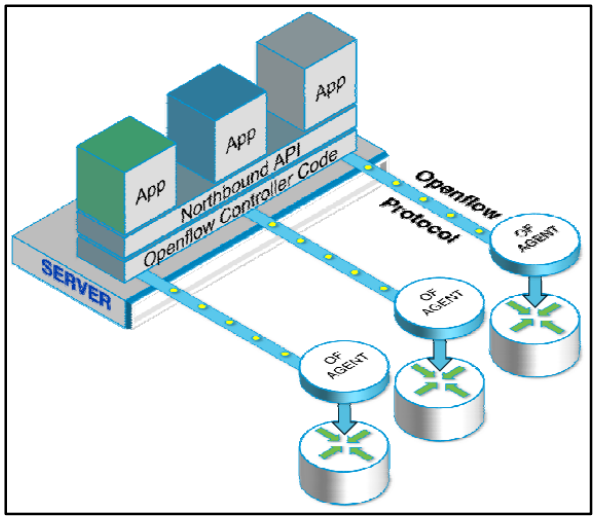


Figura 84. Esquema de comunicación mediante OpenFlow.
Fuente: (de Pozuelo, 2013). Foro Tecnológico @asLAN – Avanzando hacia las SDN. Recuperado de: http://blogs.salleurl.edu/networking-and-internet-technologies/files/2013/10/openflow.png

3.5.1. PLATAFORMA DE HARDWARE

El controlador para la red SDN se implementa sobre un servidor disponible del centro de datos FICA. En el momento del despliegue se encuentra a disposición un servidor que fue parte de un proyecto de investigación previo respecto a SDN. Las características técnicas del equipo se detallan en la Tabla 12.

Tabla 12.

Características del servidor controlador SDN.

Características técnicas del servidor	
Fabricante	Hewlett Packard
Modelo	ML150 Gen 5
Versión	1.0
Dimensiones	61,7x42,4x20cm
Tipo de Chasis	Torre – 5U
Procesador	Intel Xeon(R) CPU E5405 2.0GHz
Núcleos	4
Memoria RAM	4 GB
Disco Duro HDD	250 GB
Chip Gráfico	8 MB Compartida
Serie	MXS8460A5J
Consumo eléctrico	526 vatios
Suministro de Energía	Carga 11.6A: 100 a 127 VCA; carga: 5,5A de 200 a 240 VCA, 47 a 66 Hz
Temperatura de operación	10 a 35 ºC
Sistema Operativo	Ubuntu 16.04.3 LTS
Periféricos de Entrada y Salida	- Serie: 1 - Puntero (ratón, PS2): 1 - Gráficos: 1 - Teclado (PS2): 1 - USB 2.0: 8 (4 posteriores, 2 panel frontal, 2 internos) -RJ-45 (Ethernet): 1 (10/100/1000 Gbit/s)

3.5.2. INSTALACIÓN DE OPENDAYLIGHT

En este apartado se incluye de manera explicativa el proceso de instalación del controlador OpenDaylight, que será el encargado de tomar el control de la red SDN creada mediante la integración con el ambiente cloud. Para poner en marcha el software es necesaria la descarga de los paquetes desde la web de ODL.

La versión de OpenDaylight usada durante este despliegue es Nitrogen, que es la más actual hasta la fecha. Para obtener una copia del software nos dirigimos a sitio web del proyecto ODL indicada en la Figura 85. En la sección *Downloads* encontraremos los paquetes y su respectiva documentación alojada en una wiki con la información de la distribución más reciente, como el archivo de las versiones anteriores de la distribución.

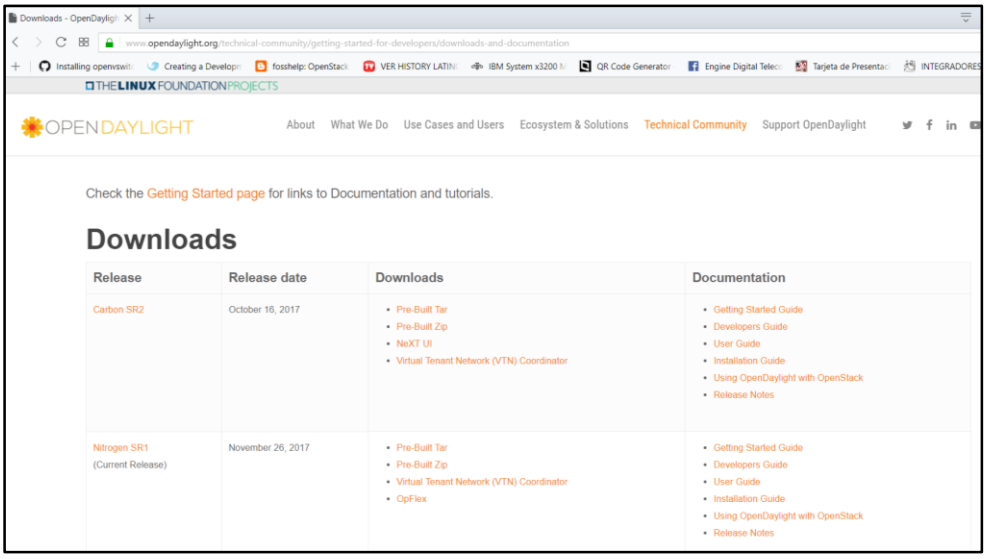


Figura 85. Sitio web de descarga de OpenDaylight.
Fuente: (OpenDaylight**,** 2017**)**.

El sistema operativo instalado en el servidor es Ubuntu Server 16.04.3 LTS. Según la documentación OpenDaylight es compatible y corre perfectamente sobre él. Uno de los pasos previos a la instalación es asegurarse de que el servidor cuenta con una IP estática, que se tiene acceso a internet y que los ficheros de resolución se encuentran correctamente configurados.

En caso de que se haya omitido algún paso durante la puesta en marcha del hardware, se corrige inmediatamente para prevenir algún error involuntario. Los parámetros del servidor deben encontrarse en el mismo grupo de comunicación de los servidores de cloud y deben estar establecidos como indica la Figura 86.

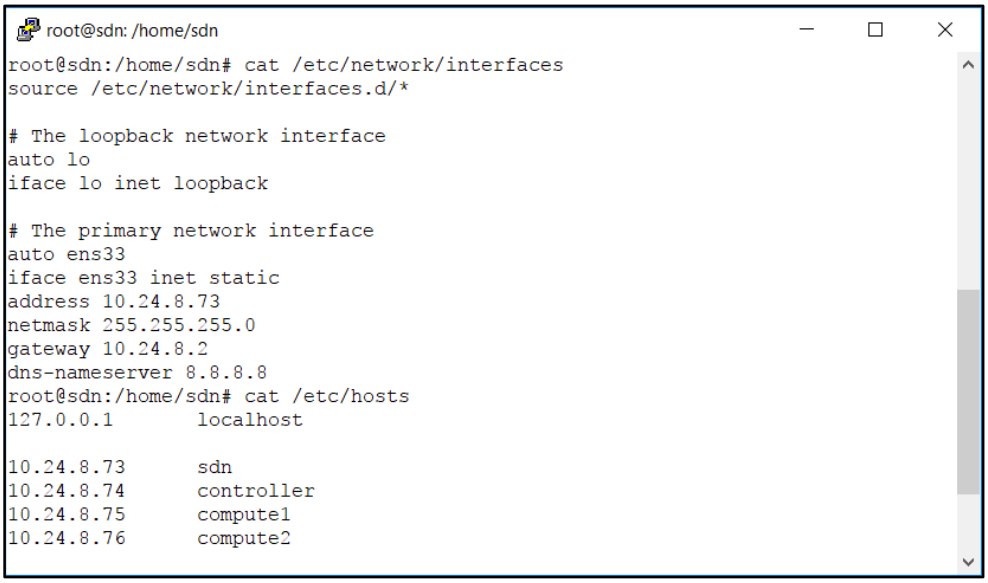


Figura 86. Parámetros de los ficheros del servidor ODL.

Los siguiente es la instalación de ciertas dependencias para el arranque de ODL, estas se encuentran alojadas en los repositorios del sistema operativo. Para obtener estos paquetes nos dirigimos a la línea de comandos y digitamos la solicitud de descarga mediante las instrucciones *add-apt-repository* y *apt-get install* seguida de los nombres de los paquetes siguientes:

- Repositorio de Oracle Java 8 (ppa:webupd8team/java)
- Java Development KIT versión 8 (oracle-java8-installer)

Una vez que inicia la instalación del paquete de Oracle Java como muestra la Figura 87, se abrirá un cuadro de dialogo que solicita la confirmación de la copia de paquetes y la aceptación de los términos de licencia del software. Cuando este proceso termina el sistema está listo para arrancar el núcleo del controlador ODL, también denominado Karaf.

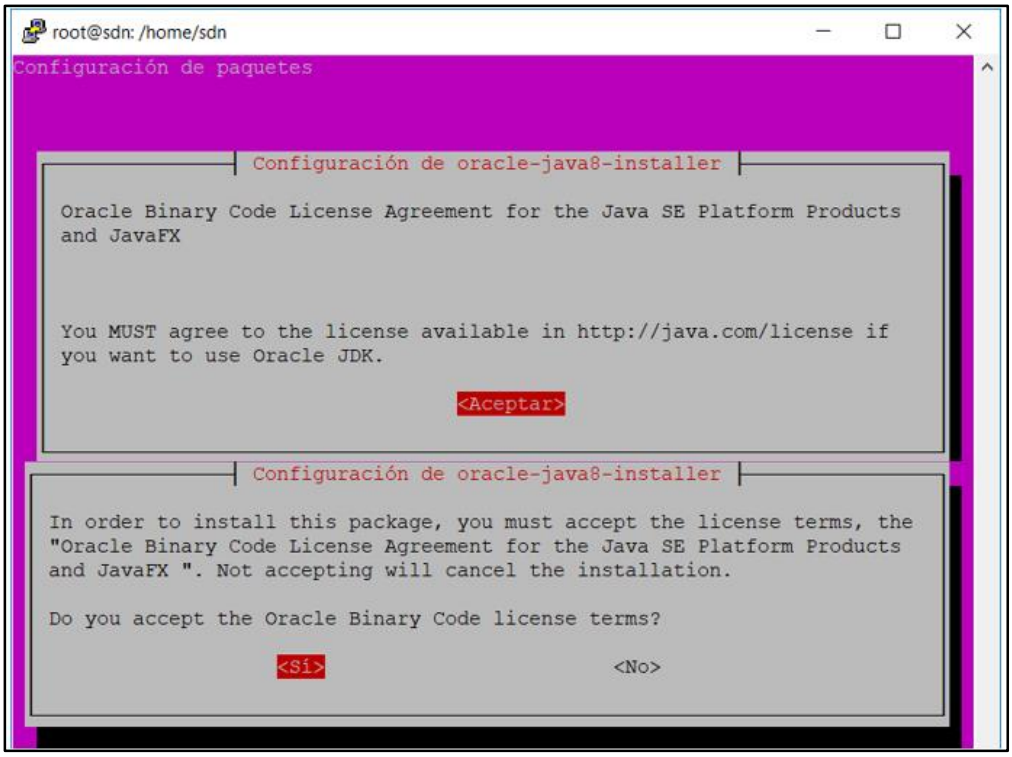


Figura 87. Términos de licencia Oracle Java 8.

Luego de instalado el paquete de desarrollo de Java, es necesario dirigirse al directorio donde se encuentre el paquete de OpenDaylight para extraer el contenido. Cuando termina la extracción se creará una carpeta conteniendo todos los archivos que hacen funcionar al software controlador. En este punto es posible visualizar que existen otros directorios con ejecutables para distintos propósitos, pero el que se debe usar para iniciar ODL es el que tiene el nombre *karaf* dentro de la carpeta /*bin*. El resultado si la ejecución se llevó a cabo de manera satisfactoria se muestra en la Figura 88, en donde se aprecia el nombre del controlador y la consola de configuración de Karaf.

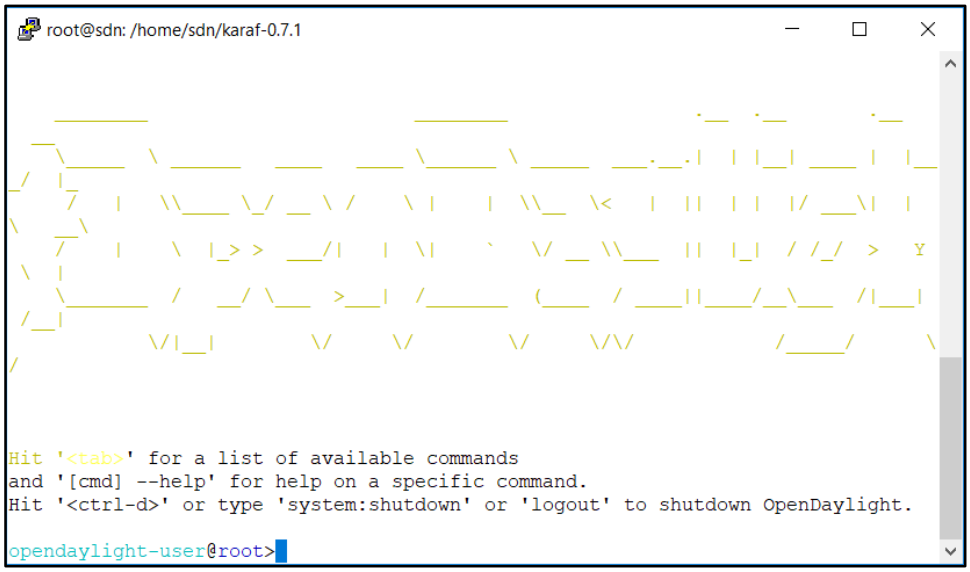


Figura 88. Consola de configuración de OpenDaylight.

3.5.3. MODULOS NETVIRT PARA OPENDAYLIGHT

En la distribución por defecto de OpenDaylight no viene habilitado ningún módulo aparte de los necesarios para el arranque y la ejecución de la consola de configuración, por lo que se debe realizar la adición de los elementos que se usará para controlar la SDN. Por razones de compatibilidad, no es posible habilitar todos los módulos simultáneamente.

Debido a que el despliegue consiste en el uso de una red definida por software de carácter hibrido, es necesario habilitar el módulo de reenvío en capa dos, los nódulos de aplicación para la integración de OpenFlow y la interfaz gráfica DLUX. Las instrucciones para habilitar estos elementos se pueden observar en la Figura 89.

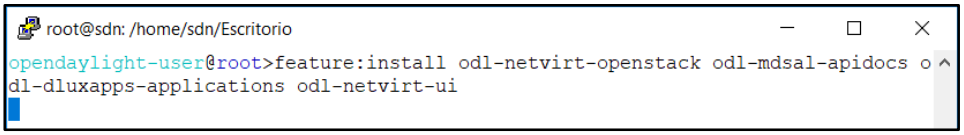


Figura 89. Instalación de módulos sobre OpenDaylight.

Para comprobar que los módulos se han instalado correctamente sobre OpenDaylight, este nos proporciona una herramienta para listar los componentes activos y las dependencias de cada uno de los módulos que se encuentran habilitados en el controlador. Al ejecutar la instrucción *feature:list –i* en la consola de configuración de Karaf se obtendrá un resultado similar al de la Figura 90.

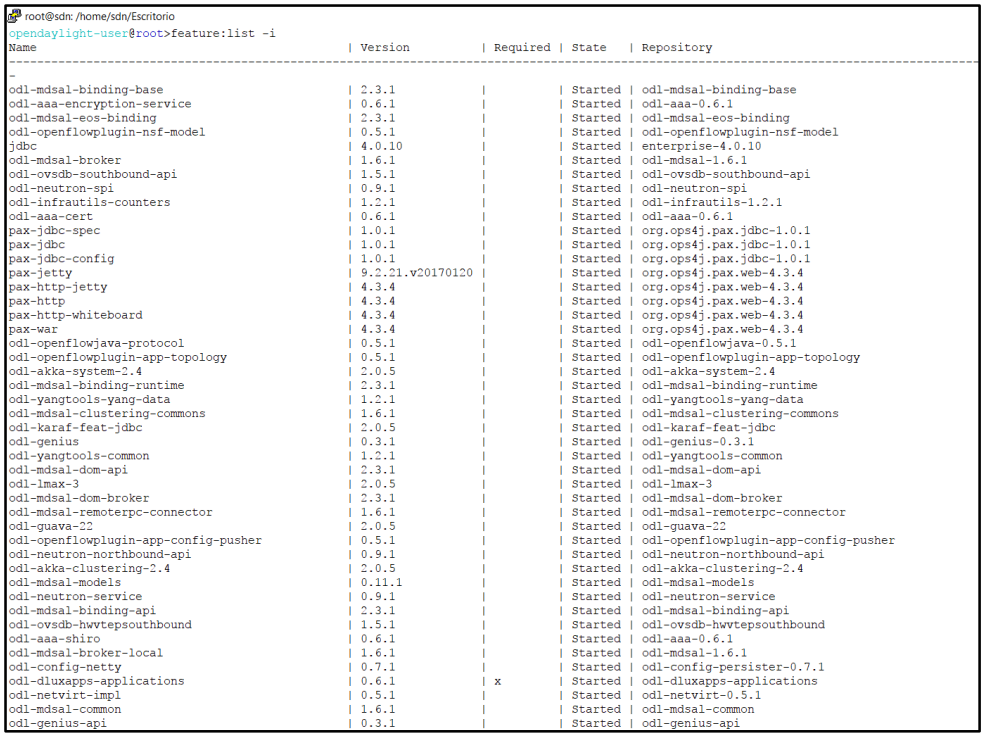


Figura 90. Módulos de NetVirt instalados en OpenDaylight

El siguiente paso para poner en funcionamiento el controlador es asegurarse que se usa la versión de Openflow más reciente, para ello es conveniente detener Karaf mientras se realiza la configuración. OpenDaylight incorpora en sus últimas versiones la capacidad de soportar tanto la versión 1.0 como la 1.3. Debido a que el software Open vSwitch usado como Back-end[13] para la

[13] Motor de procesamiento para ciertas funciones y servicios de un sistema.

red del entorno cloud trae habilitado por defecto la opción de comunicaciones mediante OpenFlow 1.3, se debe configurar el fichero *custom.properties* ubicado en el directorio *<odl folder>/etc/*, en la línea *ovsdb.of.version=1.3* con lo que se asegura que no existan inconsistencias en los mensajes de sesión entre ODL y los nodos pertenecientes al entorno de virtualización. Lo siguiente es volver a iniciar Karaf y esperar a que se recargue la configuración para poder ingresar a la interfaz DLUX con los nuevos componentes cargados como se observa en la Figura 91.

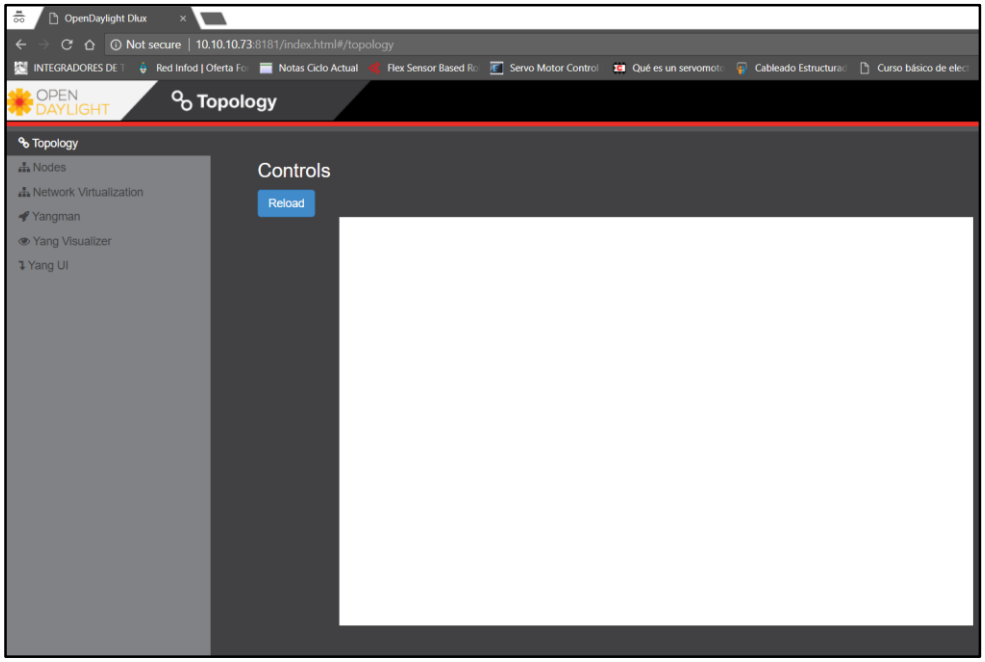


Figura 91. Entorno gráfico de los módulos NetVirt en DLUX.
Fuente: OpenDaylight SDN Controller.

3.5.4. INSTALACIÓN DEL AGENTE DE RED ODL

Para el proceso de instalación del nuevo agente de red es necesario seguir una serie de pasos en un cierto orden ya que de otra manera se ocasionarían inconsistencias en los servicios de OpenStack que dependen de Neutron y sus componentes relacionados. En primer lugar, es

imprescindible asegurar que al usar OpenDaylight como back-end de Neutron, ODL sea la única fuente real para las configuraciones de Neutron. Debido a esto es necesario eliminar las configuraciones existentes de OpenStack para proporcionar a OpenDaylight una plataforma limpia sobre la que trabajar usando las instrucciones proporcionadas por el gestor de infraestructura sobre la línea de comandos.

Se borran las instancias.

nova list

nova delete <instance names>

Se eliminan los enlaces de las subredes y los routers.

neutron subnet-list

neutron router-list

neutron router-port-list <router name>

neutron router-interface-delete <router name> <subnet ID or name>

Se eliminan los componentes de red previamente creados

neutron subnet-delete <subnet name>

neutron net-list

neutron net-delete <net name>

neutron router-delete <router name>

Se comprueba que no existen ningún puerto creado.

neutron port-list

A partir de aquí se debe realizar las modificaciones pertinentes en ficheros del sistema por lo que se debe tener sumo cuidado de no alterar equivocadamente los archivos ya que causaría la

descomposición de los servicios de OpenStack relacionados con las redes. Mientras Neutron este administrando las instancias de Open vSwitch en los nodos de cómputo y control, OpenDaylight y Neutron pueden estar en conflicto. Para evitar problemas, apagamos el servicio Neutron y deshabilitamos permanentemente los agentes de Open vSwitch en todos los nodos para otorgarle el control a OpenDaylight a través del canal de administración de los switches.

Detenemos provisionalmente neutron-server en el nodo de control.

systemctl stop neutron-server

systemctl stop neutron-openvswitch-agent

systemctl stop neutron-l3-agent.service

systemctl stop neutron-dhcp-agent.service

systemctl stop neutron-metadata-agent

systemctl stop neutron-metering-agent

Hecho esto se remueve el agente openvswitch de neutron para instalar el agente odl

systemctl stop neutron-openvswitch-agent

systemctl disable neutron-openvswitch-agent

yum remove -y openstack-neutron-openvswitch.noarch

Es necesario blanquear completamente los virtual switch para que las nuevas tablas de flujo de netvirt se instalen correctamente y no exista ningún conflicto con las previamente creadas por ovs. Tras ejecutar los comandos para blanquear los switch virtuales se debe tener una base de datos totalmente en blanco como se observa en la Figura 92, sobre la que OpenDaylight podrá escribir los nuevos parámetros de administración de los virtual switch.

service openvswitch stop

*rm -rf /var/log/openvswitch/**

rm -rf /etc/openvswitch/conf.db

service openvswitch start

Figura 92. Base de Datos en Blanco de Open vSwitch

Otro requisito no por menos importante es eliminar las referencias y puertos de Open vSwitch del kernel del sistema. Para esto se usa la herramienta *ovs-dpctl*.

ovs-dpctl del-if ovs-system br-ex

ovs-dpctl del-if ovs-system br-int

ovs-dpctl del-if ovs-system br-tun

ovs-dpctl del-if ovs-system ens33

ovs-dpctl del-if ovs-system vxlan_sys_4789

ovs-dpctl show

Una de las ventajas de trabajar sobre Linux es que se puede usar variables de entorno que pueden ser invocadas mediante rutinas o instrucciones más complejas. Mediante este método se

crean dos variables para ejecutar los comandos para generar el nuevo canal de comunicación entre el switch virtual y el controlador SDN.

```
export ODL_IP=10.10.10.77
export OS_DATA_INTERFACE=ens36
```

Se debe notar que la dirección IP usada en la primera variable es la que se encuentra en la interfaz de administración del nodo, y la interfaz de la segunda variable es la que se encuentra en la red en la que se encuentra el plano de datos. Después de ejecutarse las instrucciones automáticamente OpenDaylight debe cargar el virtual switch en la topología de DLUX como muestra la Figura 93.

```
data_interface=$(facter ipaddress_${OS_DATA_INTERFACE})
read ovstbl <<< $(ovs-vsctl get Open_vSwitch . _uuid)
ovs-vsctl set Open_vSwitch $ovstbl other_config:local_ip=${data_interface}
ovs-vsctl set-manager tcp:${ODL_IP}:6640
ovs-vsctl list Manager
ovs-vsctl list Open_vSwitch
```

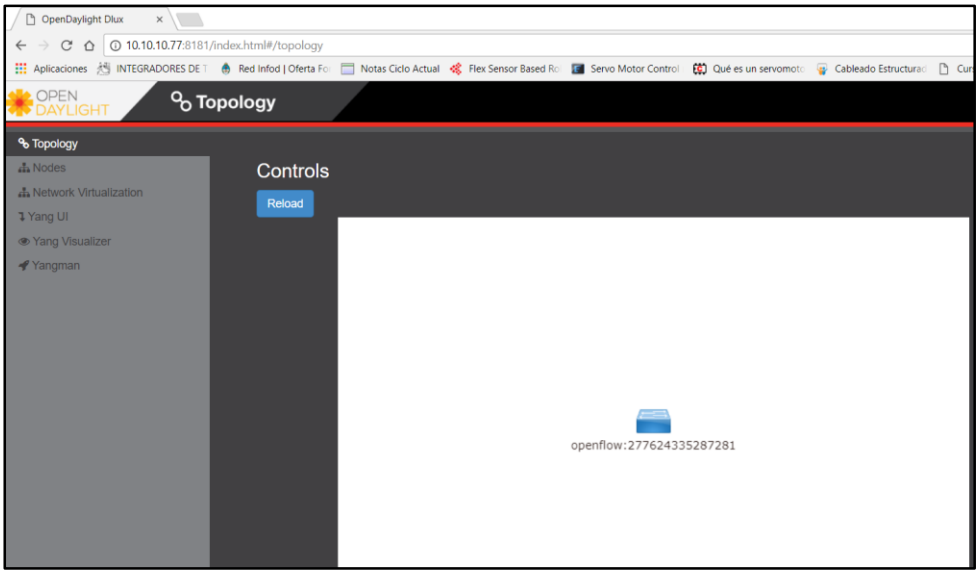


Figura 93. Virtual Switch de OpenStack cargado en OpenDaylight
Fuente: OpenDaylight SDN Controller.

El siguiente paso es verificar y asegurarse que la interfaz br-ex existe y se encuentra correctamente configurada con el puerto relacionado a la interfaz física. Para ello usamos una instrucción con el binario *cat* para recuperar la información de los ficheros de red involucrados. El resultado debe ser como el de la Figura 94.

```
root@controller:~
[root@controller ~]# cat /etc/sysconfig/network-scripts/ifcfg-br-ex
DEVICE=br-ex
DEVICETYPE=ovs
TYPE=OVSBridge
BOOTPROTO=static
IPADDR=10.24.8.74
NETMASK=255.255.255.0
GATEWAY=10.24.8.2
ONBOOT=yes
PEERDNS=yes
PEERROUTES=yes
[root@controller ~]# cat /etc/sysconfig/network-scripts/ifcfg-ens33
DEVICE=ens33
TYPE=OVSPort
DEVICETYPE=ovs
OVS_BRIDGE=br-ex
ONBOOT=yes
[root@controller ~]#
```

Figura 94. Configuración de las interfaces del bridge de acceso externo

Neutron cuenta con un fichero relacionado con la forma en que se provee el servicio de DHCP y algunos parámetros dedicados a los distintos escenarios que actualmente se soportan. NetVirt requiere que se modifiquen un par de campos para proporcionar DHCP correctamente a los hosts haciendo uso de Open vSwitch.

nano /etc/neutron/dhcp_agent.ini

[DEFAULT]

force_metadata = True

[OVS]

ovsdb_interface = vsctl

El módulo de capa 2 también cuenta con un fichero de configuración, en donde se deben incluir los parámetros respecto a los drivers de red que se usaran, en este caso VXLAN, así como el mecanismo de gestión de estos drivers que para NetVirt es OpenDaylight. También es indispensable proporcionarle a Open vSwitch una url para las transacciones a través de la API.

nano /etc/neutron/plugins/ml2/ml2_conf.ini

[ml2]

type_drivers = flat,vxlan

tenant_network_types = vxlan

mechanism_drivers = opendaylight

[ml2_odl]

password = admin

username = admin

url = http://10.10.10.77:8181/controller/nb/v2/neutron

port_binding_controller = pseudo-agentdb-binding

Para que todos los cambios en los ficheros sean cargados por Neutron y OpenDaylight es necesario reconstruir la base de datos, que debido a las modificaciones se ha vuelto inconsistente. Para recrear la base de datos ejecutamos las instrucciones desde adentro del gestor de base de datos de MariaDB[14] en donde se deberá crear publicar las nuevas entradas en el gestor de base de datos de Neutron con el motivo de que OpenStack normalice su funcionamiento ahora usando a *OpenDaylight como único controlador de los servicios de red de la plataforma.*

```
mysql -e "drop database if exists neutron;"
mysql -e "create database neutron character set utf8;"
mysql -e "grant all on neutron.* to 'neutron'@'%';"
su -s /bin/sh -c "neutron-db-manage --config-file /etc/neutron/neutron.conf \
--config-file /etc/neutron/plugins/ml2/ml2_conf.ini upgrade head" neutron
```

Finalmente instalamos networking-odl que no es más que una biblioteca de controladores y complementos que integra OpenStack Neutron API con OpenDaylight backend. Por ejemplo, tiene el controlador ML2 y el complemento L3 para permitir la comunicación de la API de recursos OpenStack Neutron L2 y L3 al Backend OpenDaylight. Una vez que se ha instalado se reinician todos los servicios de red de OpenStack para crear los enlaces de los dispositivos virtuales en ODL.

```
yum -y install python-networking-odl
systemctl start neutron-server
neutron-odl-ovs-hostconfig
```

[14] Sistema de gestión de bases de datos derivado de MySQL con licencia GPL.

systemctl start neutron-dhcp-agent.service

systemctl start neutron-metadata-agent

systemctl start neutron-metering-agent

systemctl start neutron-l3-agent.service

Para comprobar que la integración se ha llevado a cabo con éxito se debe abrir una sesión ssh en el servidor principal de OpenStack y tras obtener privilegios con la llave generada por Keystone se ejecuta la instrucción para listar los agentes de red, en donde ahora aparece el agente de capa 2 perteneciente a OpenDaylight como muestra la Figura 95. Esto confirma que el agente ODL se encuentra integrado con OpenStack.

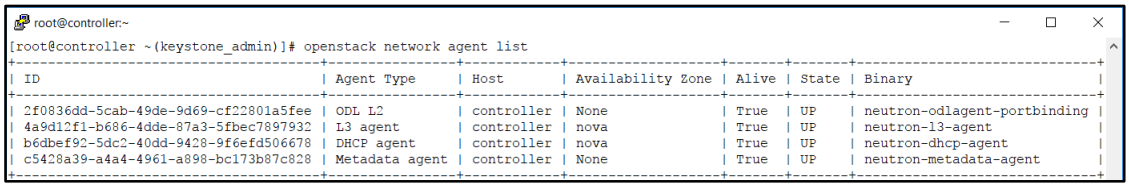


Figura 95. Agente de Red ODL L2

CAPITULO IV

4. PRUEBAS DE FUNCIONAMIENTO, ANALISIS DE RESULTADOS

En este capítulo se realizan las pruebas realizadas sobre el ambiente desplegado, que validan a este proyecto como resultado al problema definido en el capítulo 1, y finalmente se comentan los resultados y las conclusiones del presente proyecto, así como el futuro trabajo que se puede desarrollar sobre la misma línea de investigación.

4.1. VERIFICACIÓN DE TRÁFICO

Cuando OpenDaylight se integra con OpenStack automáticamente se muestran los switches y los conectores de cada virtual switch dentro de la pestaña Nodes. Para el caso del controlador de la plataforma, muestra inicialmente un solo conector llamado br-int como se aprecia en la Figura 96. Este es el puente que se encarga de interconectar todos los dispositivos virtualizados en OpenStack usando los recursos almacenados en el datastore de OpenDaylight y las tablas de flujo predeterminadas por NetVirt. De esta manera se construye de manera mecánica el canal de comunicación Openflow de Este a Oeste, es decir de tráfico horizontal.

Node Id	Node Name	Node Connectors	Statistics
openflow:181390869806952	None	1	Flows | Node Connectors

Node Connector Id	Name	Port Number	Mac Address
openflow:181390869806952:LOCAL	br-int	4294967294	a4:f9:5a:5c:3f:68

Node Connector Id	Rx Pkts	Tx Pkts	Rx Bytes	Tx Bytes	Rx Drops	Tx Drops	Rx Errs	Tx Errs	Rx Frame Errs	Rx OverRun Errs	Rx CRC Errs	Collisions
openflow:181390869806952:LOCAL	0	0	0	0	0	0	0	0	0	0	0	0

Figura 96. Bridge Principal y conector local de OpenStack
Fuente: OpenDaylight SDN Controller.

Toda la información mostrada en las tablas se refleja en la interfaz de usuario de NetVirt que es capaz de representar los puentes de red y los componentes sintácticos a través de una topología

sencilla compuesta de switches y bridges virtuales. Cada uno de estos elementos es capaz de proporcionar información de sus características y funciones dentro de la red, así como las tablas de flujo que manejan. La Figura 97 muestra el bridge principal en su estado inicial, mientras el datastore de OpenDaylight se encuentra en blanco.

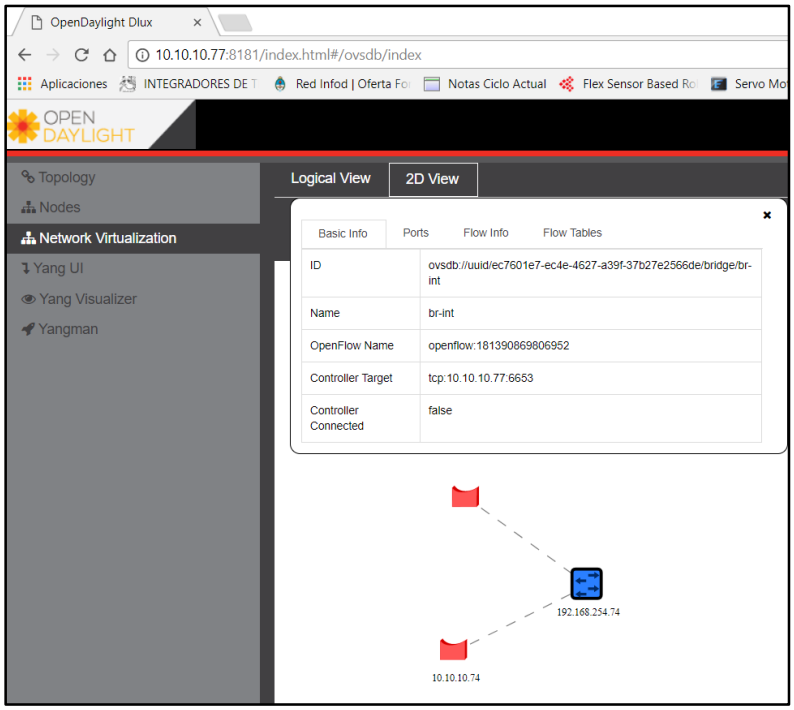


Figura 97. Representación de br-int en la Interfaz NetVirt
Fuente: OpenDaylight SDN Controller.

4.1.1. TABLAS DE FLUJO DE NETVIRT

La respuesta de las tablas de flujo de Open vSwitch antes de la integración con OpenDaylight se encontraban casi en blanco ya que el flujo de información era como en un switch normal. La implementación del agente ODL de capa 2 hace que una serie de flujos se instalen en las tablas de los Switch Virtuales para crear el ambiente y establecer el comportamiento de una red virtual. El contenido de las tablas de flujo después de la integración se puede observar en la Figura 98 en donde se diferencian distintas clases de flujos, entre las que se hallan los clasificadores (Tabla 0)

y los procesadores de ARP (Responders). Para generar las tablas de flujo de OVS se usa la instrucción *ovs-ofctl -OOpenFlow13 dump-flows br-int*.

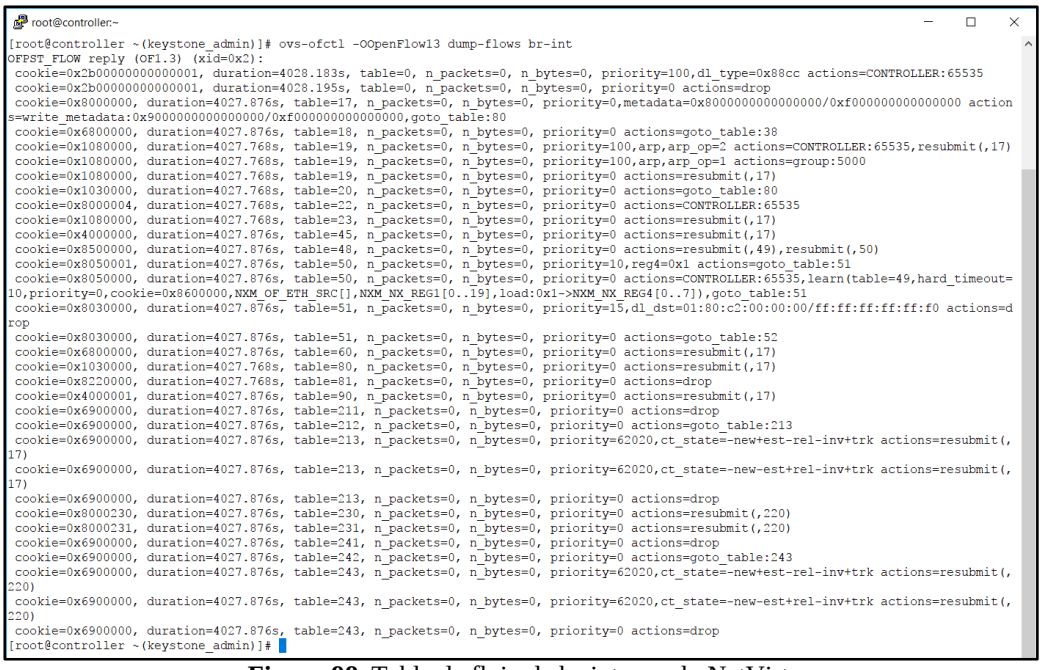


Figura 98. Tabla de flujo de br-int usando NetVirt.

Debido a que NtVirt cuenta con los plugins *netvirtneutronnorthbound, OVSDBSouthbound Plugin, OpenFlowSouthbound Plugin y HWVTEP Southbound Plugin*, NetVIrt puede controlar los dispositivos virtuales y los dispositivos de hardware a través del protocolo de administración OVSDB de Open vSwitch mediante la creación de tablas dinámicas con flujos reactivos basados en su propia pipeline o cadena de funciones.

4.1.1.1. Consideraciones de la Tabla (0)

La tabla cero consiste en el clasificador de paquetes el cual discrimina entre tres tipos de paquetes: los emitidos localmente, los enviados desde otros nodos, y los dirigidos al controlador. Estos paquetes pueden ser clasificados de acuerdo a su coincidencia o match con alguna de las alternativas.

- Match en el paquete enviado localmente: De acuerdo a la condición IN_PORT y la MAC del host se identifica si el paquete es local y se agrega el ID del túnel. Se hace un match entre la MAC y el puerto conectado mas no en la dirección IP relacionada. Además de la adición del ID del túnel re configura un registro de metadatos para diferenciar el paquete de otros generados en otros hosts.

- Match con los paquetes de otros nodos: Se identifica paquetes de otros nodos basándose en la ID de túnel y la condición IN_PORT para averiguar el puerto de origen, luego de la misma manera se agrega información de metadatos para su transporte.

- PACKET_IN: Se envían al controlador los paquetes LLDP y se procesan por el complemento de administración de topología. No se hace manejo especial a los mensajes de ARP y DHCP.

4.1.1.2. Consideraciones de la Tabla (20)

Esta tabla es la encargada del procesamiento de las solicitudes ARP, usando el ARP Responser, OVS Reply y ARP Reply. La Figura 99 presentada a continuación muestra el principio de funcionamiento de una respuesta ARP usando Openflow.

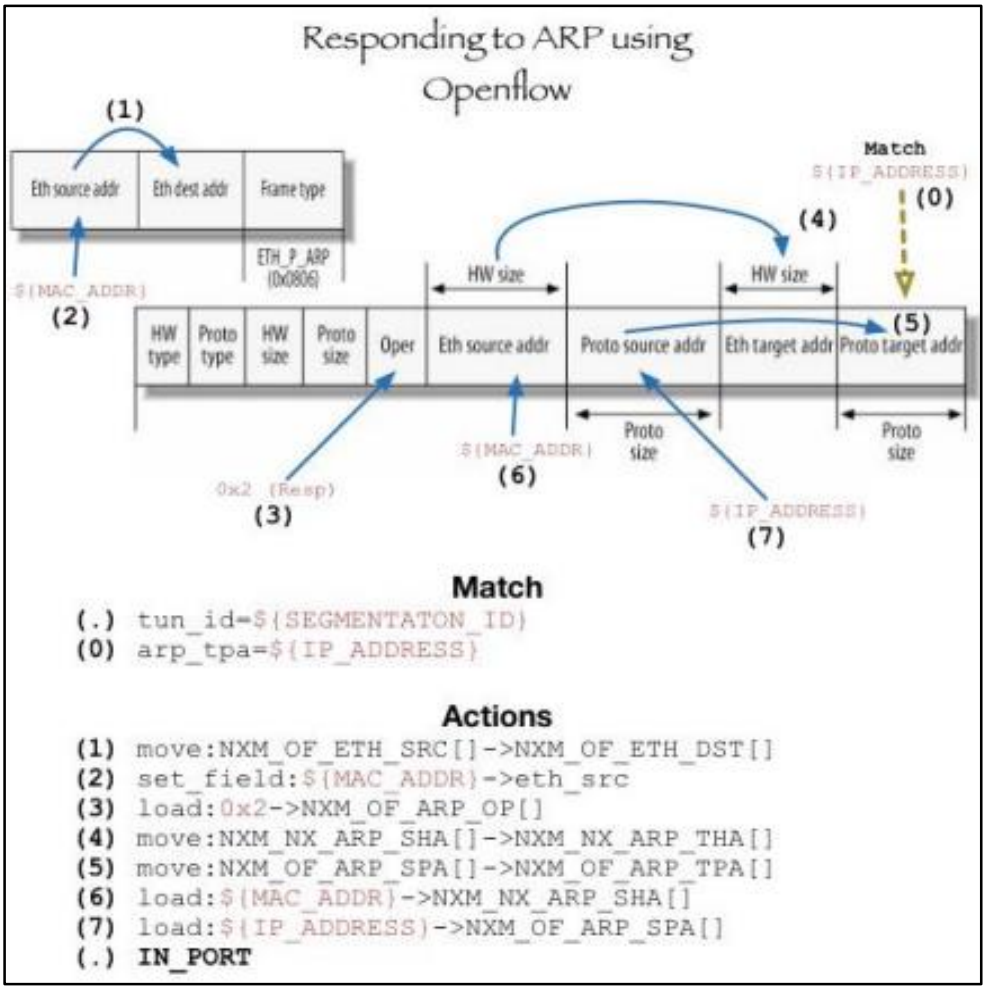


Figura 99. Respuesta ARP usando Openflow.
Fuente: (Xining, 2016). Análisis de la tabla de flujo de Netvirt. Recuperado de: http://www.sdnlab.com/17506.html

Los tipos de paquete que tienen respuestas ARP pueden ser enviadas desde los host o desde los puertos DHCP virtuales, las solicitudes de ARP también pueden ser enviadas por nodos remotos y puertos remotos, en caso de usarse la característica de reenvío de capa 3 las interfaces de los router realizan solicitudes de ARP para registrarse en las tablas de flujo y las IP flotantes al ser generadas como puertos virtuales también realizan solicitudes ARP para anunciar su existencia en la topología de red.

4.1.1.3. Consideraciones de la Tabla (60)

Esta tabla contiene las entradas de las máquinas virtuales locales que se encuentran en la tabla de rutas de todas las redes alcanzables en el tenant formando una malla completa. Cada vez un tenant crea una subred se agregan reglas n*(n-1) en todos los hosts donde se encuentra este tenant. A diferencia de Neutron donde se pueden crear múltiples enrutadores, solo hay un vrouter en un tenant de NetVirt.

La ventaja de esta implementación es que, para el tráfico de este a oeste, la consulta solo se debe realizar una vez. El rendimiento del controlador se garantiza ya que no es necesario pasar por él. Sin embargo, para topologías complejas es difícil administrar y mantener todas las tablas de flujo de manera manual.

4.1.2. CREACIÓN DE COMPONENTES DE RED VIRTUALES

En esta sección se da lugar a la creación de todos los elementos que conforman la red virtual dentro de la plataforma desplegada. Entre los componentes más importantes se encuentran la red, la subred de aprovisionamiento y su respectivo router de acceso. El router se conecta a la red externa y al mismo tiempo a la red lógica creada.

El componente de OpenStack que se encarga de virtualizar la red es Neutrón, un componente que ha sido sometido a un importante desarrollo y ha ido incluyendo paulatinamente múltiples tecnologías de red como backends mediante plugins. En este despliegue Open vSwitch se utiliza como la tecnología de capa 2 por debajo de Neutron para realizar la virtualización de la red usando un esquema como el de la Figura 100.

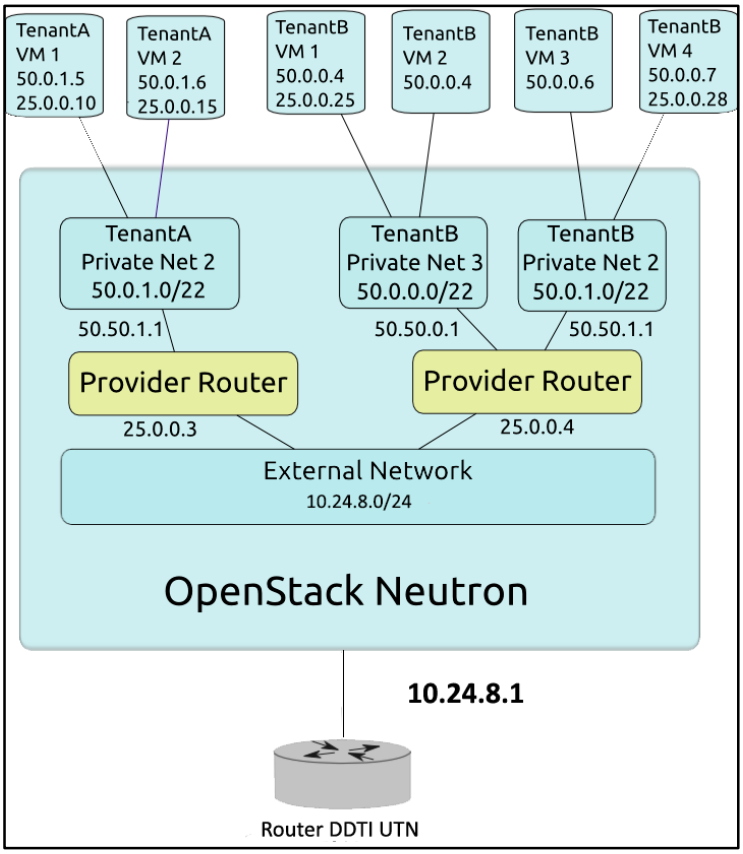


Figura 100. Esquema de virtualización para la red del Cloud FICA.
Fuente: (Emergya, 2016**)**. Curso OpenStack. Recuperado de: http://iesgn.github.io/emergya/curso/u8/openvswitch

4.1.3. REDES PRIVADAS

Puesto que el despliegue completo incorpora una gran gama de elementos y la explicación se vuelve un tanto compleja, se detalla en primer lugar la configuración de capa 2 con Open vSwitch y posteriormente se prosigue con la parte que involucra la creación de los elementos que funcionan en la capa 3.

4.1.3.1. Bridge de Integración

La finalidad de la virtualización de una red es lograr que las instancias de una red común se puedan comunicar entre sí a través del uso del mismo direccionamiento y compartiendo algunos elementos comunes como el servicio DHCP y la puerta de enlace a la vez que se mantengan

aisladas de otras redes virtuales. Este comportamiento debe suceder en cada hipervisor o nodo de cómputo en el que se ejecuten las máquinas virtuales. Una de las formas de conseguir esto es con Open vSwitch mediante la creación de un switch virtual en cada nodo denominado switch de integración o *br-int* y comunicar cada nodo con los demás mediante túneles VXLAN que encapsulan el tráfico entre los nodos tal como se indica en la Figura 101.

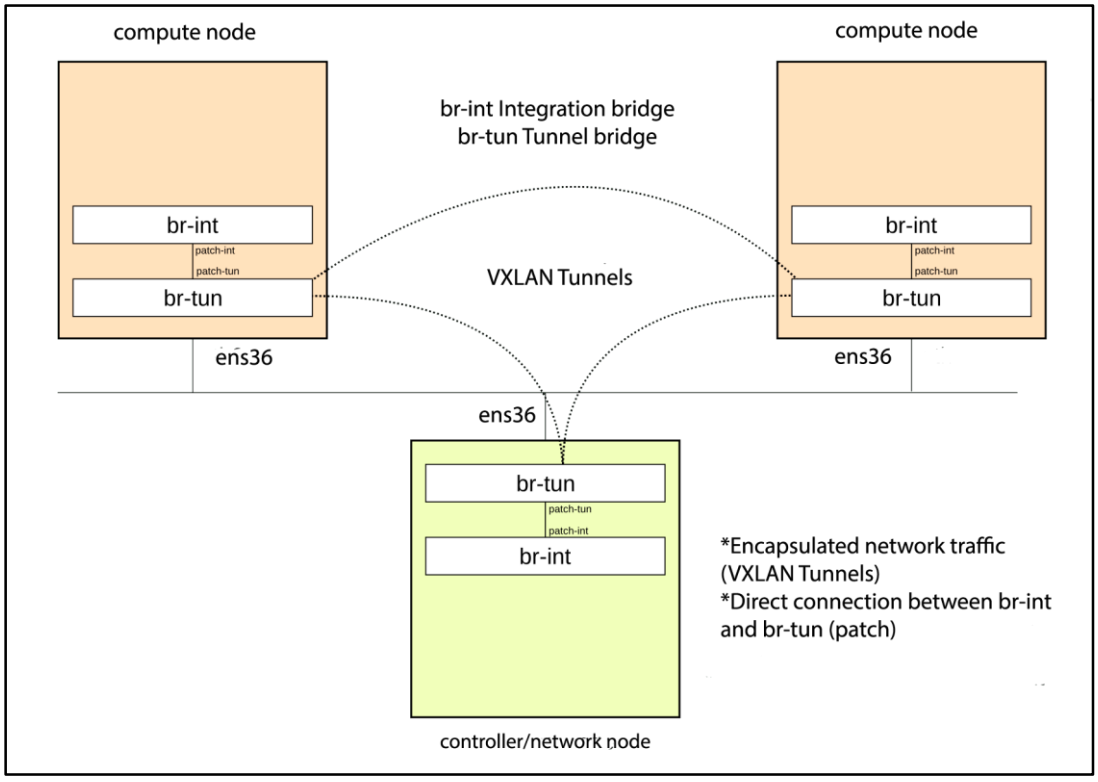


Figura 101. Esquema de conexión de los bridges virtuales.
Fuente: (Emergya**,** 2016**)**. Curso OpenStack. Recuperado de: http://iesgn.github.io/emergya/curso/u8/openvswitch

Puesto que se va a detallar el proceso de creación de algunos bridges y puertos de conexión paso a paso, es necesario mostrar cual es el estado inicial del sistema antes de configurar. Inicialmente los únicos bridges existentes son el br-ex creado inicialmente para proporcionar conectividad con la red externa y el *br-int* generado automáticamente durante la integración con OpenDaylight el cual muestra una composición como la que se muestra en la Figura 102.

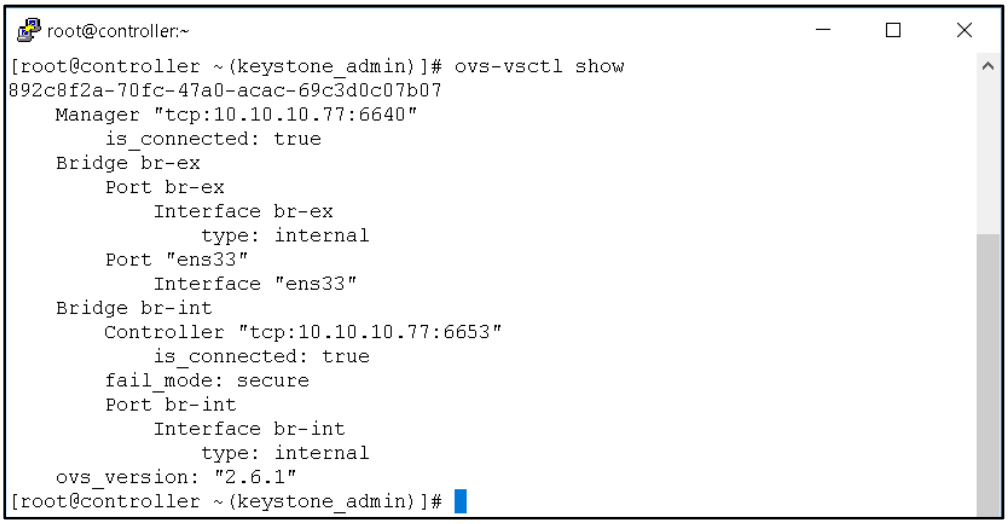


Figura 102. Bridges virtuales inicialmente en blanco.

Inicialmente no existe ningún bridge físicamente creado sobre Open vSwitch.

```
[root@controller ~(keystone_admin)]# brctl show
bridge name     bridge id               STP enabled     interfaces
```

En el nodo que se encuentra instalado neutron-server existen inicialmente dos puentes de Open vSwitch, en este apartado se trabaja principalmente con el bridge de integración.

```
[root@controller ~(keystone_admin)]# ovs-vsctl list-br
br-ex
br-int
```

La única interfaz presente en br-int es LOCAL debido a que el switch virtual acaba de ser creado, mediante la instrucción *ovs-ofctl –Oopenflow13 show br-int* podemos recuperar la información sobre el soporte y características con las que cuenta el puente de red creado a base de Open vSwitch.

```
[root@controller ~(keystone_admin)]# ovs-ofctl -OOpenflow13 show br-int
OFPT_FEATURES_REPLY (OF1.3) (xid=0x2): dpid:00005c65a9ddec68
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS GROUP_STATS
QUEUE_STATS
OFPST_PORT_DESC reply (OF1.3) (xid=0x3):
 LOCAL(br-int): addr:5c:65:a9:dd:ec:68
    config:     PORT_DOWN
    state:      LINK_DOWN
    speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (OF1.3) (xid=0x5): frags=normal miss_send_len=0
```

Usando el cliente de línea de comandos de Neutrón y con las credenciales de un usuario cualquiera, se ejecuta la creación de una red privada y una subred asociada a ella con el rango de direcciones IP de prueba, que en este caso es 10.0.0.0/24. En la Figura 103 se muestra el retorno de las instrucciones que se detallan a continuación. Estas pueden ser ejecutadas múltiples veces para crear varias redes, pero se debe tomar en cuenta que cada ID de red es muy distinto y la asociación con cada subred debe ser relacionada con la red correcta.

```
[root@controller ~]# . keystonerc_admin
[root@controller ~(keystone_admin)]# neutron net-create privada
[root@controller ~(keystone_admin)]# neutron subnet-create <ID DE RED> 10.0.0.0/24
```

Figura 103. Red virtual tipo VXLAN creada con Neutron.

La red anteriormente definida no se crea realmente hasta que no se lanza una instancia para la cual se crean todos los dispositivos de red necesarios y se interconectan adecuadamente. Para completar el proceso de creación de la red creamos una instancia desde la línea de comandos o desde la interfaz gráfica de usuario asegurándonos de usar la nueva red como fuente de

aprovisionamiento. En la Figura 104 se puede observar la nueva instancia usando una dirección del rango anteriormente asignado.

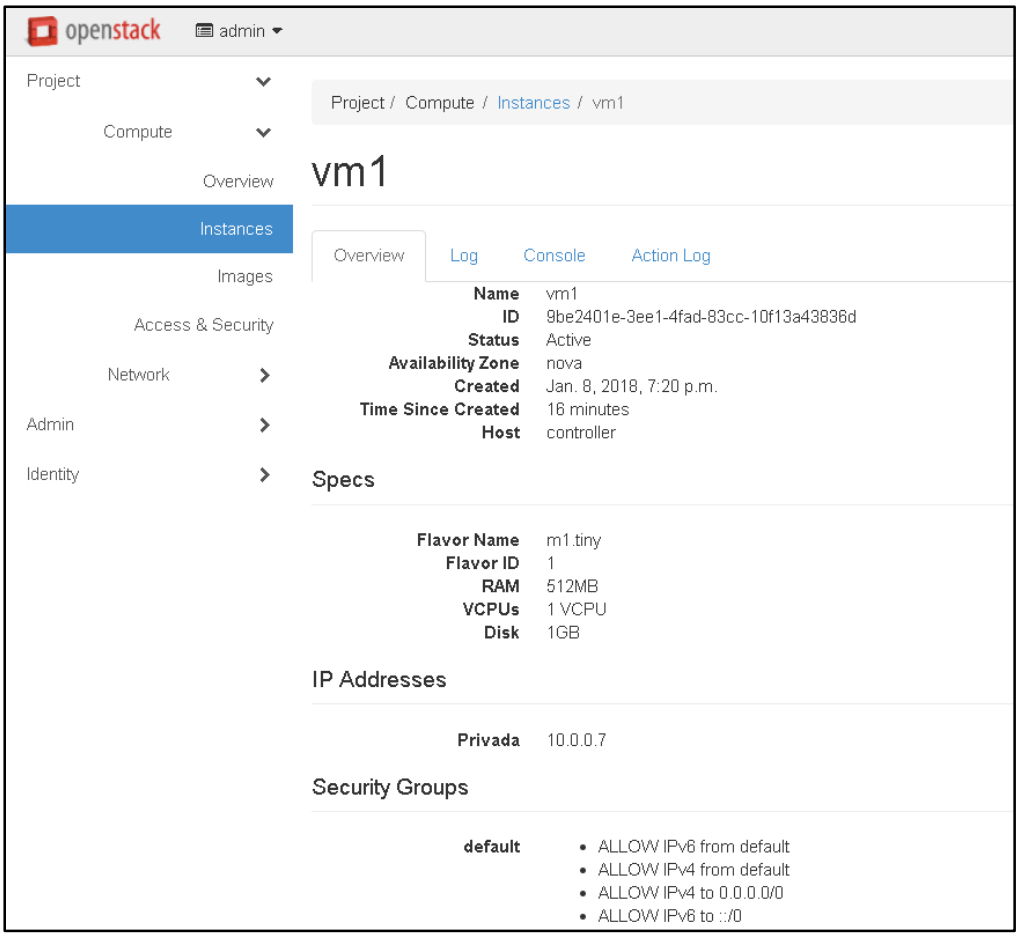


Figura 104. Instancia aprovisionada mediante la red virtual.

En este punto comprobamos que se han creado dos puertos nuevos en la nueva subred. Para mostrar la lista de todos los puertos usamos la instrucción de lista de Neutron y filtramos el resultado por la subred anteriormente creada. El retorno puede variar dependiendo de la cantidad de redes que se encuentren creadas en el momento de ejecutar el comando.

```
[root@controller ~(keystone_admin)]# neutron port-list|grep 6a5a33bc-20b8-497d-85f2-3ca5d31dbf7b

| 7e1afbb9-e8e4-45c6-8452-33bfc234f2b2 |      | fa:16:3e:5e:8f:08 | {"subnet_id": "6a5a33bc-20b8-497d-85f2-3ca5d31dbf7b", "ip_address": "10.0.0.2"} |

| c841a540-1b39-4a41-a1f2-6400879169e2 |      | fa:16:3e:41:24:ae | {"subnet_id": "6a5a33bc-20b8-497d-85f2-3ca5d31dbf7b", "ip_address": "10.0.0.7"} |
```

Es posible comprobar también cuál de los puertos corresponde con el servidor DHCP y cual con la instancia que se ha creado. A la vez que revisamos el retorno de esta instrucción a través de la línea de comandos, podemos justificar también toda la información que Neutron aloja en el datastore de OpenDaylight desde la interfaz de usuario en el apartado de NetVirt como se muestra en la Figura 105. Aquí se hallará el esquema lógico de los elementos que se han añadido a través de Neutron.

```
[root@controller ~(keystone_admin)]# neutron port-show 7e1afbb9-e8e4-45c6-8452-33bfc234f2b2|grep device_owner

| device_owner          | network:dhcp                                                    |

[root@controller ~(keystone_admin)]# neutron port-show c841a540-1b39-4a41-a1f2-6400879169e2|grep device_owner

| device_owner          | compute:nova                                                    |
```

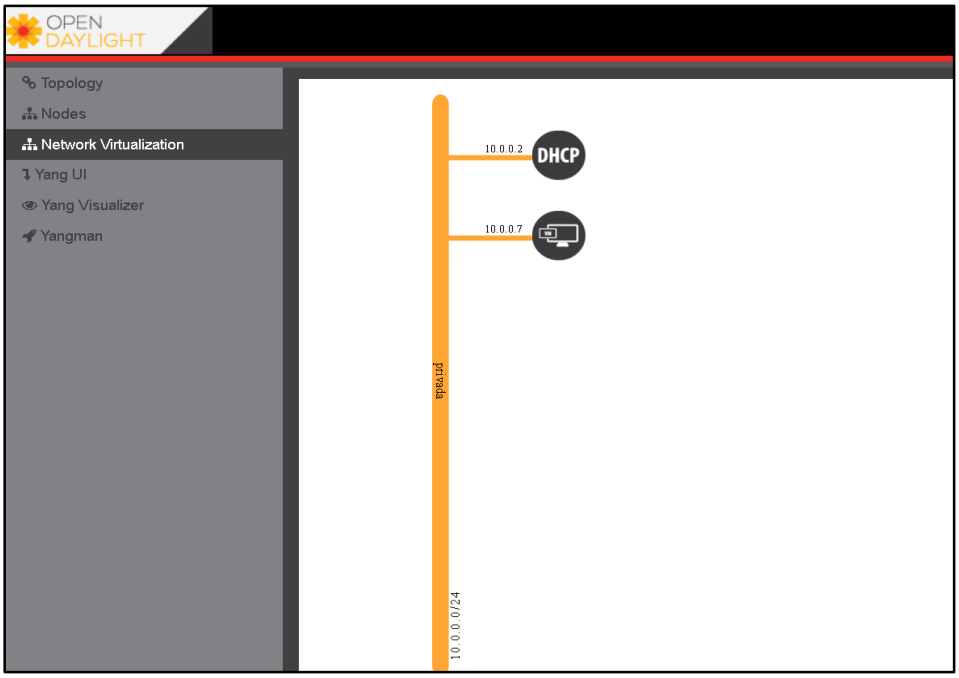


Figura 105. Representación lógica de la red de OpenStack en OpenDaylight
Fuente: OpenDaylight SDN Controller.

De la información obtenida se entiende que el puerto con la IP 10.0.0.2 corresponde al servidor DHCP y el otro con la instancia recién lanzada. Estos puertos también aparecen en el bridge de integración por lo que usando el comando *ovs-ofctl -OOpenFlow13 dump-flows br-int* se mostraran las estadísticas de los puertos de br-int.

```
[root@controller ~(keystone_admin)]# ovs-ofctl -OOpenFlow13 show br-int
OFPT_FEATURES_REPLY (OF1.3) (xid=0x2): dpid:0000a418c14d18de
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS GROUP_STATS
QUEUE_STATS
OFPST_PORT_DESC reply (OF1.3) (xid=0x3):
 1(tap7e1afbb9-e8): addr:00:00:00:00:e0:00
     config:     PORT_DOWN
     state:      LINK_DOWN
     speed: 0 Mbps now, 0 Mbps max
```

```
2(tapc841a540-1b): addr:fe:16:3e:41:24:ae
    config:     0
    state:      0
    current:    10MB-FD COPPER
    speed: 10 Mbps now, 0 Mbps max
 LOCAL(br-int): addr:a4:18:c1:4d:18:de
    config:     PORT_DOWN
    state:      LINK_DOWN
    speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (OF1.3) (xid=0x5): frags=normal miss_send_len=0
```

La denominación de los puertos en Open vSwitch está relacionada con el identificador único de puerto en OpenStack, concretamente con los 11 primeros caracteres. Esto puede resultar bastante difícil de entender, sobre todo en la diferenciación de los elementos implicados, pero gracias a OpenDaylight su interfaz amigable facilita la lectura de la topología de red.

4.1.3.2. Servicio DHCP

Este servicio utiliza Linux Network Namespaces, un modo de virtualización de la red en donde se puede usar el mismo identificador varias veces en diferentes espacios de nombres. También permite la restricción del uso de los identificadores dependiendo de los procesos que soliciten acceso o uso. En el nodo principal es posible ejecutar la instrucción *ip netns* para obtener los UUID[15] de cada espacio de nombres.

```
[root@controller ~(keystone_admin)]# ip netns
qdhcp-2adb5378-96e0-4443-bc58-9e26172ec96e
```

[15] Identificador Único Universal conformado por 32 dígitos hexadecimales usado para identificar el mismo elemento en diferentes contextos.

Ejecutando la instrucción *ip netns* se puede observar que ha sido creado un espacio de nombres que empieza por dhcp, se puede ejecutar comandos dentro de este espacio de nombres de manera independiente sin ninguna relación con el sistema y el resultado será como el de un dispositivo linux con interfaces totalmente propias.

Accediendo al directorio con el nombre del identificador para el servicio DHCP y visualizando el contenido del fichero hosts, se puede observar que existe reservas MAC para las instancias que se van creando. La Figura 106 muestra una consulta al fichero en el que se realizan las reservas MAC por cada dirección IP de la red creada y sus respectivas replicas en el datastore de OpenDaylight.

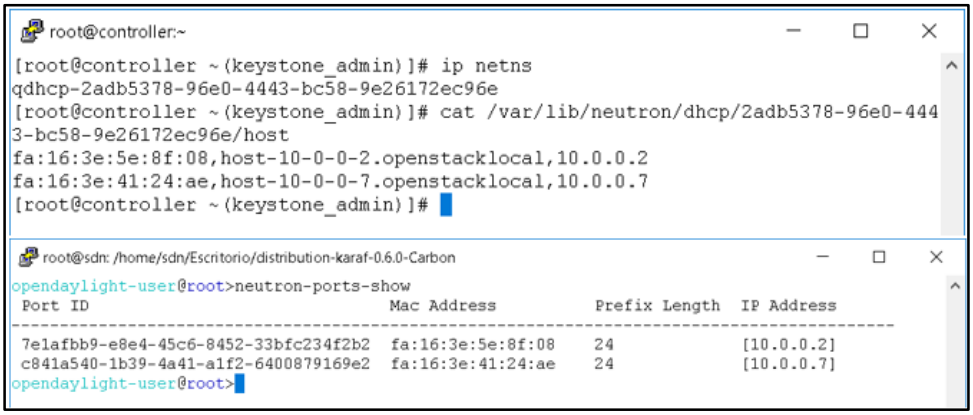


Figura 106. Reserva MAC para las instancias de OpenStack.

4.1.4. RED EXTERNA

En la sección anterior se ponen en marcha los mecanismos que permiten la conectividad entre instancias que se ejecutan en el mismo o diferentes nodos de computación, así como el aislamiento entre diferentes redes mediante el uso de los namespaces. En esta sección se explica de forma detallada el proceso que se sigue para conectar las instancias al exterior mediante el uso de la misma red virtual.

Partiendo de la situación inicial en la que el bridge exterior ya se encuentra creado en el servidor donde se halla neutron-server y con la interfaz externa anclada al puente de red, se analiza el estado de este bridge y como fluye el tráfico a través de él. La instrucción *ovs-ofctl show br-ex* permite recuperar todas las estadísticas de la interfaz que es parte de Open vSwitch.

```
[root@controller ~(keystone_admin)]# ovs-ofctl show br-ex
OFPT_FEATURES_REPLY (xid=0x2): dpid:0000000c29733299
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS
ARP_MATCH_IP
actions: output enqueue set_vlan_vid set_vlan_pcp strip_vlan mod_dl_src
mod_dl_dst mod_nw_src mod_nw_dst mod_nw_tos mod_tp_src mod_tp_dst
 1(ens33): addr:00:0c:29:73:32:99
     config:     0
     state:      0
     current:    1GB-FD COPPER AUTO_NEG
     advertised: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER
AUTO_NEG
     supported:  10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER
AUTO_NEG
     speed: 1000 Mbps now, 1000 Mbps max
 LOCAL(br-ex): addr:00:0c:29:73:32:99
     config:     0
     state:      0
     speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0
```

Esta situación se representa esquemáticamente en la Figura 107, en la que se puede observar que el bridge exterior se encuentra conectado a la interfaz ens33 y se encuentra desconectado de los otros dos bridges que hacen parte de las redes internas, incluyendo los túneles entre nodos que se crean al añadir más hipervisores.

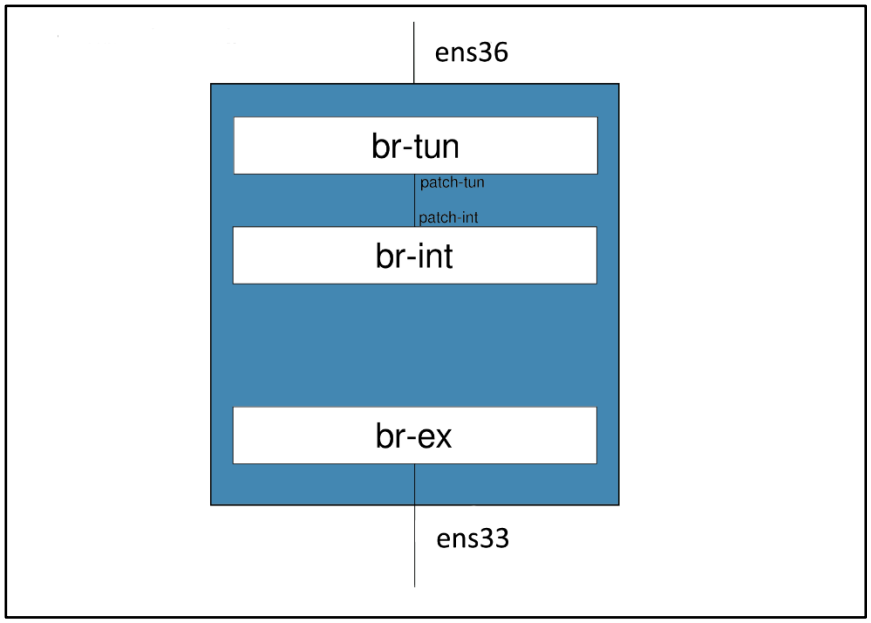


Figura 107. Esquema lógico del bridge externo.

La dirección IP del bridge externo es 10.24.8.74 y está en una red 10.24.8.0/24 de la que no se dispone totalmente ya que es el pool de direcciones de la DMZ de la Universidad Técnica del Norte. Por esta razón el número de IP flotantes será bastante limitado, pero de ninguna manera afectará a la creación de la red externa.

Se debe tomar en cuenta que la creación de redes externas solo se le permite al administrador de la plataforma por lo que se debe usar las credenciales proporcionadas por el gestor de infraestructura. Luego se ingresan las instrucciones para la creación de la red y posteriormente se crea una subred asociada usando los datos dela red física que permitirá el acceso a las instancias de OpenStack, haciendo uso del bridge de integración controlado mediante Openflow y el bridge externo conectado a una red plana. La Figura 108 muestra el retorno de los comandos usados para la creación de una red externa.

Figura 108. Creación de una red externa.

Se puede ingresar en Horizon con un usuario cualquiera y se podrá observar en la sección Topología de Red que ha aparecido una nueva red no gestionada por el usuario. Por un lado, se tiene las redes privadas gestionadas por cada proyecto y una red exterior gestionada por el administrador. El único elemento que resta por añadir para conectar las redes privadas con el exterior es un enrutador.

4.1.5. ROUTER VIRTUAL

En el despliegue de OpenStack que se está utilizando cada proyecto puede definir sus propias redes privadas y enrutadores, estos router son los dispositivos que permiten conectar las redes privadas con la red exterior y realmente lo que van a hacer es conectar el bridge de integración con el bridge exterior. Como usuario normal, se define un nuevo router para conectar la red privada con el exterior. La salida será como en la Figura 109.

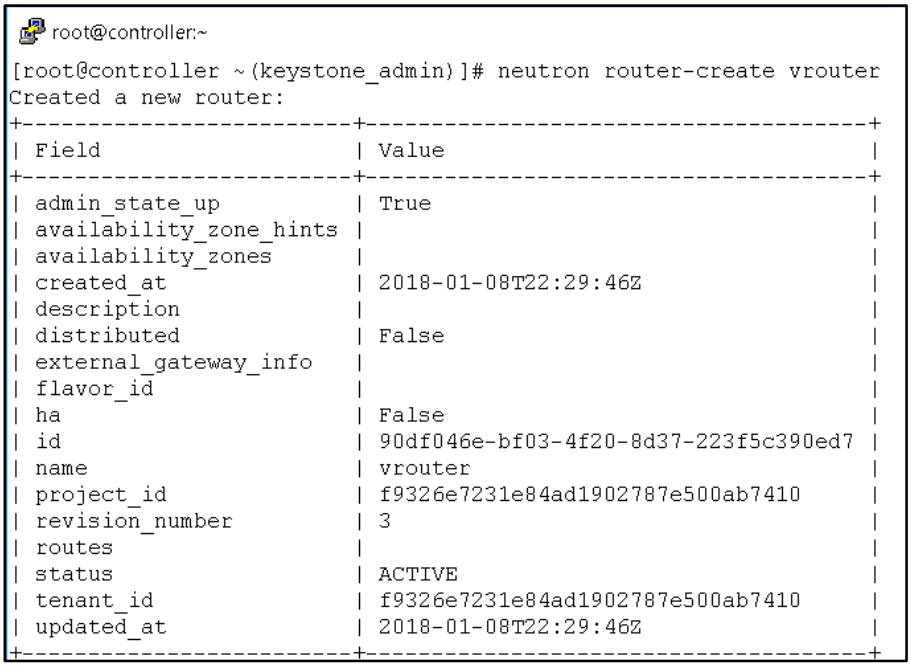


Figura 109. Creación del router virtual.

Conectamos el router a la red exterior, lo que se denomina en Neutron como configurar la puerta de enlace.

```
[root@controller ~(keystone_admin)]# neutron router-gateway-set 90df046e-bf03-4f20-8d37-223f5c390ed7 2f438218-17e3-4af7-bfc0-da9ddb46f213
Set gateway for router 90df046e-bf03-4f20-8d37-223f5c390ed7
```

Los router también se crean en un espacio de nombres de red propio, por lo que se puede comprobar con la instrucción *ip netns* que aparece un nuevo espacio de nombres. Si vemos las direcciones IP de las interfaces de red en ese espacio de nombres que aparece en la Figura 110, se comprueba la existencia de una interfaz con prefijo "qg- " seguido de 11 caracteres asociados al nuevo puerto creado.

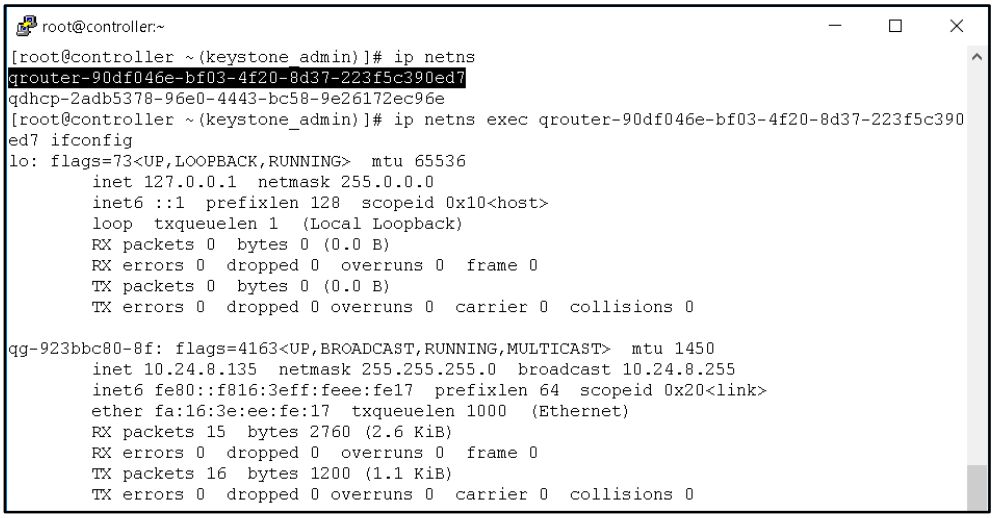


Figura 110. Espacio de nombres del router virtual.

Si comprobamos con las credenciales de administrador de la plataforma los puertos activos, se puede ver que aparece un nuevo puerto asociado a la primera dirección IP del rango configurado para la red externa. En este caso la dirección que se asigna a la interfaz outside del router virtual es 10.24.8.135 lo que significa que todo se encuentra en orden y desde el espacio de nombres será posible alcanzar cualquier destino como se muestra en la Figura 111.

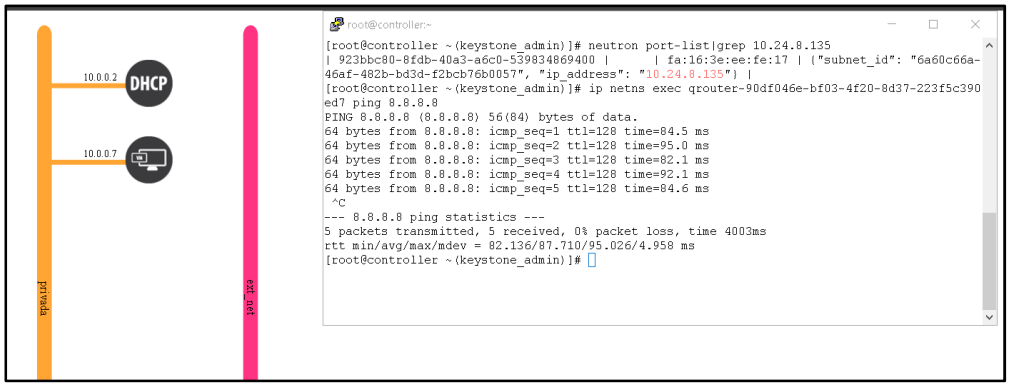


Figura 111. Espacio de Nombres del router virtual con acceso externo.

A pesar de que todo se encuentra correctamente configurado, OpenDaylight puede no mostrar el router ya que este es un dispositivo totalmente OpenFlow por lo que únicamente se mostrará la red externa. En el caso de que las tablas Openflow se actualicen de inmediato y las interfaces se encuentren conectadas al dispositivo el resultado de la creación de un virtual router será como muestra la figura 112.

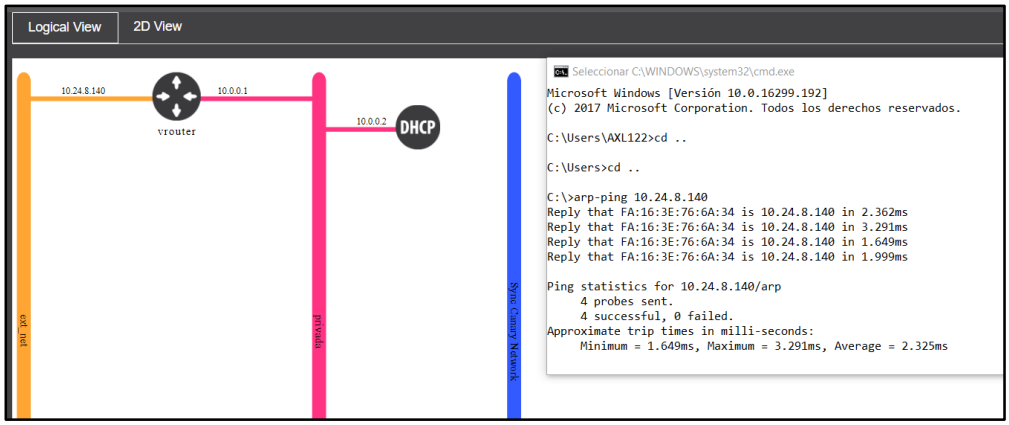


Figura 112. Router basado en reglas Openflow

Ahora, para conectar la red privada es necesario conocer el ID de la subred y se ejecuta la instrucción sobre el router virtual.

```
[root@controller ~(keystone_admin)]# neutron router-interface-add 90df046e-bf03-4f20-8d37-223f5c390ed7 6a5a33bc-20b8-497d-85f2-3ca5d31dbf7b

Added interface 47d43f72-4f2e-4568-ab6a-f44e8f31831f to router 90df046e-bf03-4f20-8d37-223f5c390ed7.
```

Si la instrucción se ha ejecutado correctamente se puede comprobar que se ha creado un nuevo puerto en el proyecto asociado a la dirección 10.0.0.1, esto sucede debido a que por defecto se reserva la primera dirección IP de la subred especialmente para el router. En el espacio de nombres también aparece una nueva interfaz asociada a dicho puerto con la misma IP.

La denominación de las interfaces agregadas a un router sigue el mismo criterio de los casos anteriores, aunque en este caso se usa el prefijo "qr- " y posteriormente los 11 caracteres del puerto con el que se relaciona. Esta interfaz de red se encuentra conectada al bridge de integración, como puede verificarse con el comando *ovs-ofctl show br-int|grep <nombre de la interfaz>*. La Figura 112 muestra la información de todos los puertos necesarios para conectar las instancias al exterior.

```
root@controller:~
[root@controller ~(keystone_admin)]# neutron port-list
+--------------------------------------+------+-------------------+--------------------------------------+
| id                                   | name | mac_address       | fixed_ips                            |
+--------------------------------------+------+-------------------+--------------------------------------+
| 47d43f72-4f2e-4568-ab6a-f44e8f31831f |      | fa:16:3e:64:cd:b9 | {"subnet_id": "6a5a33bc-20b8-497d-85f2-3ca5d31dbf7b", "ip_address": "10.0.0.1"} |
| 7e1afbb9-e8e4-45c6-8452-33bfc234f2b2 |      | fa:16:3e:5e:8f:08 | {"subnet_id": "6a5a33bc-20b8-497d-85f2-3ca5d31dbf7b", "ip_address": "10.0.0.2"} |
| 923bbc80-8fdb-40a3-a6c0-539834869400 |      | fa:16:3e:ee:fe:17 | {"subnet_id": "6a60c66a-46af-482b-bd3d-f2bcb76b0057", "ip_address": "10.24.8.135"} |
| c841a540-1b39-4a41-a1f2-64008 79169e2 |      | fa:16:3e:41:24:ae | {"subnet_id": "6a5a33bc-20b8-497d-85f2-3ca5d31dbf7b", "ip_address": "10.0.0.7"} |
+--------------------------------------+------+-------------------+--------------------------------------+
[root@controller ~(keystone_admin)]# ovs-ofctl -OOpenflow13 show br-int|grep qr-47d43f72-4f
 3(qr-47d43f72-4f): addr:00:00:00:00:70:d9
[root@controller ~(keystone_admin)]#
```

Figura 113. Puertos creados en Open vSwitch para conectar las instancias con el exterior.

4.2. CLUSTERIZACIÓN DE LA PLATAFORMA

La última etapa de la implementación es la clusterizacion de los dos nodos restantes con las mismas características del nodo de control. Este proceso consta de dos partes, la primera es el aprovisionamiento desde el nodo principal inyectando los scripts de configuración a los nodos esclavos y la segunda la reconfiguración de los ficheros relacionados con el servicio Neutron para que se enlacen con el controlador SDN usando la interfaz OVSDB de la plataforma NetVirt.

4.2.1. ADICIÓN DE NODOS ESCLAVOS

Desde el nodo principal se modifica el fichero de respuestas usado por PackStack en un inicio, añadiendo las direcciones IP de los nodos esclavos y excluyendo la IP del controlador para que no se sobrescriban los ficheros ya creados. En la Figura 114 se indica la configuración del fichero para añadir los nodos esclavos al entorno cloud.

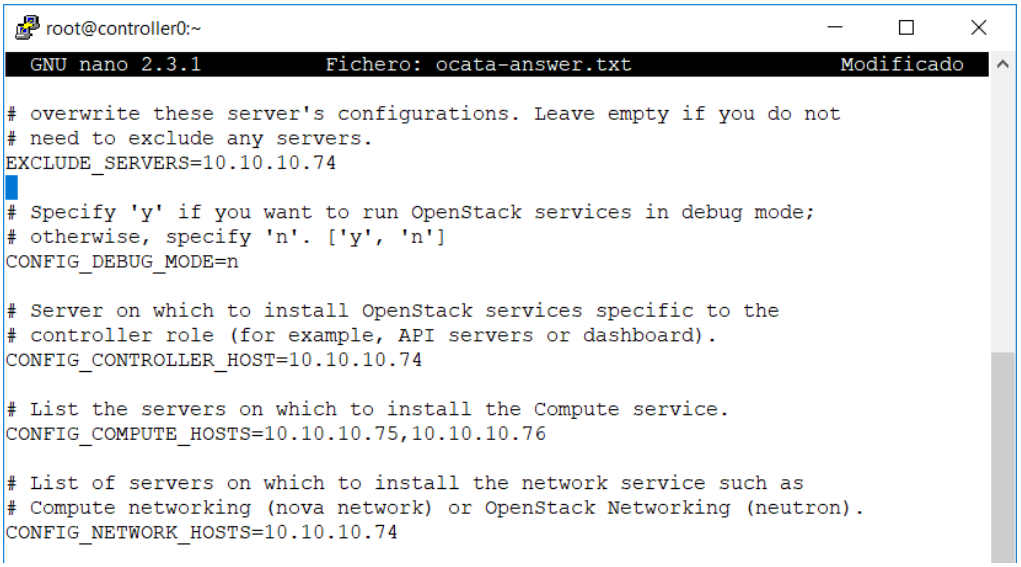


Figura 114. Configuración para añadir nodos con PackStack.

Luego de editar el fichero se arranca PackStack usando como parámetro el script de respuestas modificado, con esto únicamente se ejecuta la instalación de las herramientas en los nodos adicionales y se crean los enlaces con los servicios dependientes alojados en el nodo principal. La Figura 115 muestra una imagen durante la ejecución del proceso de adición de nodos.

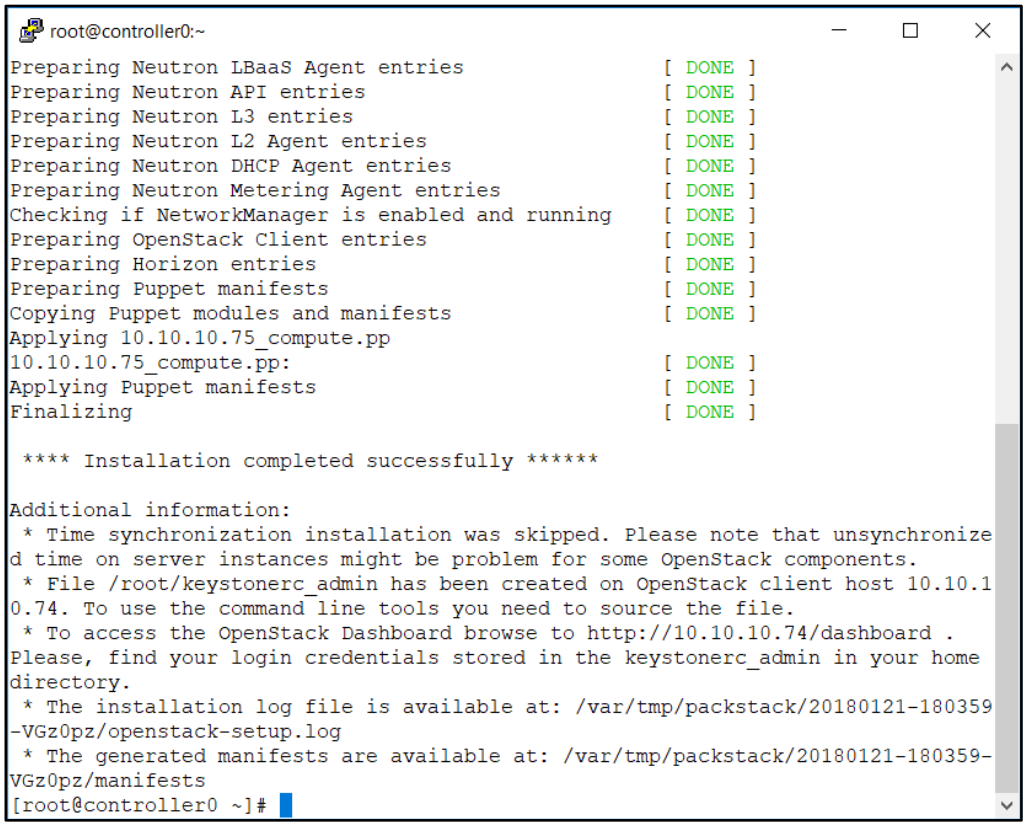


Figura 115. Adición de nodos con PackStack.

4.2.2. ENLACE DE NODOS A OPENDAYLIGHT

En este punto todos los nodos tienen habilitado el switch virtual pero no mantienen comunicación entre si y tampoco son gestionados mediante OpenFlow. Para habilitar los Open vSwitch de cada nodo para trabajar bajo el control de OpenDaylight, es necesario editar los ficheros del agente de red Neutron en forma similar al nodo principal y para ello se usan las instrucciones que se detallan a continuación.

En primer lugar, se instala el driver networking-odl que es el encargado de generar las reglas Openflow una vez que el controlador tenga el control del virtual switch de cada nodo.

yum -y install git

git clone https://github.com/openstack/networking-odl.git -b stable/ocata

cd networking-odl/

git checkout stable/ocata

python ./setup.py install

Luego se detiene el servicio Neutron, en el nodo principal y se desactiva permanentemente los agentes de Open vSwitch controlados por Neutron para permitir la instalación del nuevo agente de ODL en cada nodo.

En el nodo de control.

systemctl stop neutron-server

En los nodos esclavos.

systemctl stop neutron-openvswitch-agent

systemctl disable neutron-openvswitch-agent

Con los servicios sensibles inactivos se blanquea los virtual switch y se configuran los enlaces con OpenDaylight cuidando los parámetros de cada nodo para la correcta creación de los túneles de comunicación VXLAN. En la Figura 116 se muestra los nodos enlazados con OpenDaylight mediante la interfaz OVSDB de Open vSwitch.

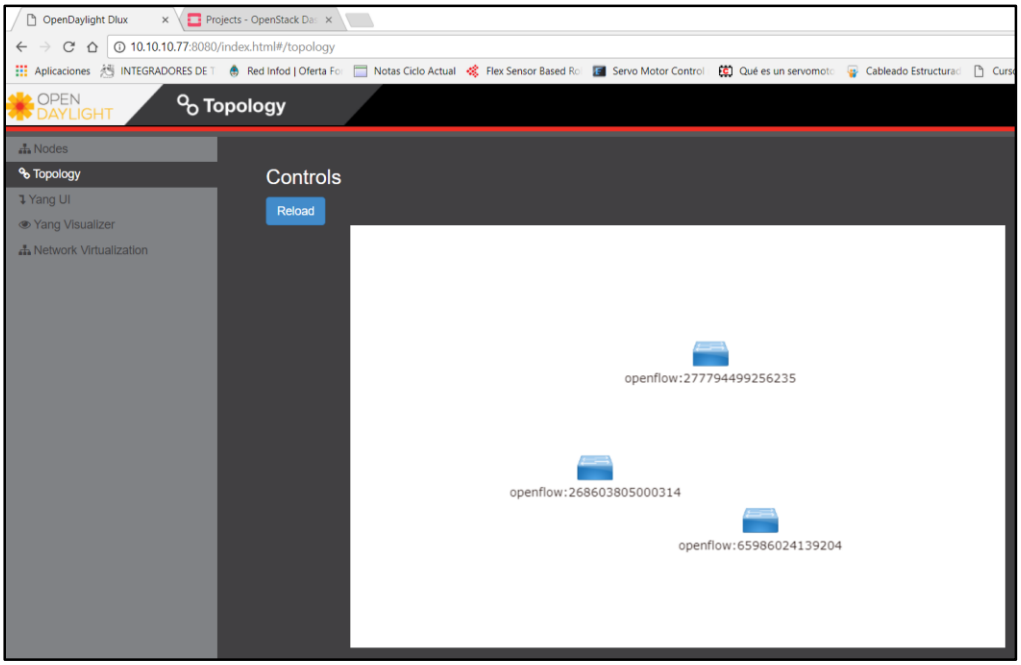


Figura 116. Miembros del clúster controlados por OpenDaylight.

systemctl stop openvswitch

*rm -rf /var/log/openvswitch/**

rm -rf /etc/openvswitch/conf.db

systemctl start openvswitch

Se elimina las referencias a los puertos existentes en Open vSwitch.

ovs-dpctl del-if ovs-system br-ex

ovs-dpctl del-if ovs-system br-int

ovs-dpctl del-if ovs-system br-tun

ovs-dpctl del-if ovs-system ens33

Se crean y se levantan las interfaces de acceso externo en los nodos esclavos.

nano /etc/sysconfig/network-scripts/ifcfg-br-ex

DEVICE=br-ex

DEVICETYPE=ovs

TYPE=OVSBridge

BOOTPROTO=static

IPADDR=10.24.8.75

NETMASK=255.255.255.0

GATEWAY=10.24.8.2

ONBOOT=yes

PEERDNS=yes

PEERROUTES=yes

nano /etc/sysconfig/network-scripts/ifcfg-ens33

DEVICE=ens33

TYPE=OVSPort

DEVICETYPE=ovs

OVS_BRIDGE=br-ex

ONBOOT=yes

ifdown br-ex

ifdown ens33

ifup ens33

ifup br-ex

Se establece OpenDaylight como controlador por defecto para los virtual switch.

ovs-vsctl set Open_vSwitch . other_config:local_ip=10.10.10.75

ovs-vsctl set Open_vSwitch . other_config:provider_mappings=extnet:br-ex

ovs-vsctl set-manager tcp:10.10.10.77:6640

Inmediatamente después de que el controlador registra los nodos en el datastore, se crean conectores para cada switch virtual con un ID único que se usa para construir los enlaces lógicos en la plataforma NetVirt, poco después de que se generan estos enlaces la topología es representada a través de la pestaña Network Virtualization en el entorno DLUX como muestra la Figura 117. Los iconos de color azul representan los Open vSwitch y los iconos de color rojo representan los bridges creados en cada virtual switch, siendo los bridges de integración los que aparecen con la dirección IP del nodo y los bridges externos aquellos que se encuentran si identificador.

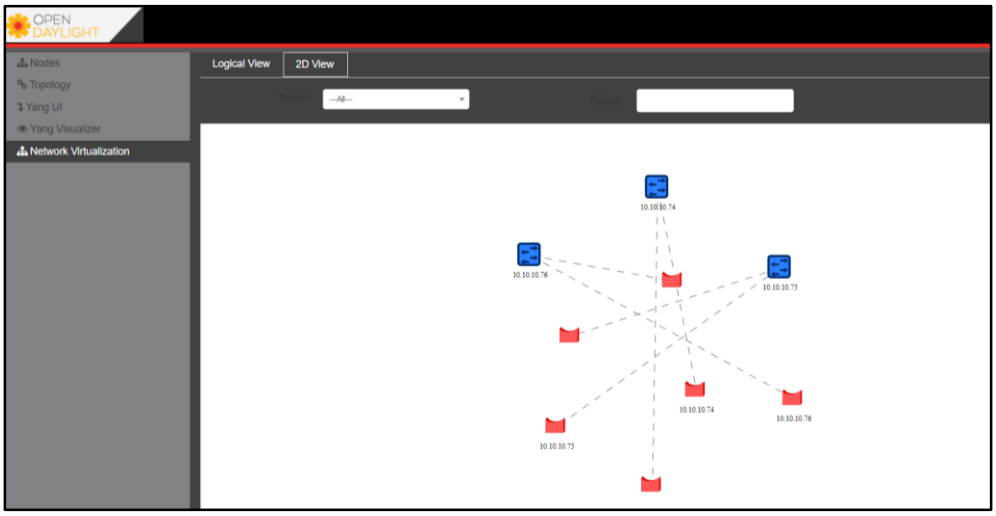


Figura 117. Topología lógica de la red del clúster.

4.3. PRUEBA DE NODOS INTEGRADOS

Para comprobar el estado de los enlaces entre nodos y del clúster de recursos se ejecuta una prueba a través de la API que proporciona OpenDaylight pudiendo usarse la línea de comandos para hacer llamadas al datastore y confirmar que los nodos se encuentran conectados a través de redes administradas por el controlador de red.

4.3.1. CONSULTAS MEDIANTE API_REST AL DATASTORE DE OPENDAYLIGHT

La API REST se encuentra disponible en la misma dirección IP del controlador SDN, pero a diferencia de la consola la API REST solo soporta el protocolo HTTP. Esta API solo admite el envío y recepción de datos cifrados en UTF-8 por lo cual se debe configurar el cliente http para codificar y decodificar los caracteres correctamente. Todas las respuestas se presentan en formato JSON y también se hallan codificadas en UTF-8.

Las primeras solicitudes que se puede realizar son las de lectura de las redes, subredes y routers creados y almacenadas en el datastore de OpenDaylight. Para esto se puede ocupar las siguientes líneas de instrucciones sobre la línea de comandos de cualquiera de los nodos. El resultado devuelto es como el que se muestra en la Figura 118.

curl -u admin:admin http://10.10.10.77:8181/controller/nb/v2/neutron/networks

curl -u admin:admin http://10.10.10.77:8181/controller/nb/v2/neutron/subnets

curl -u admin:admin http://10.10.10.77:8181/controller/nb/v2/neutron/routers

curl -u admin:admin http://10.10.10.77:8181/controller/nb/v2/neutron/ports

Figura 118. Consulta al datastore de OpenDaylight mediante REST.

Otro tipo de consultas se pueden realizar directamente desde el navegador ingresando la url que enlaza con la interfaz del datastore, permitiendo visualizar el contenido de ciertas tablas. No todas las tablas son accesibles ya que pueden constar de secciones operacionales, así como de configuración. A continuación, se indican varias url que permiten tener acceso a información del entorno NetVirt y la Figura 119 muestra el retorno en el navegador.

http://10.10.10.77:8181/restconf/operational/network-topology:network-topology/

http://10.10.10.77:8181/restconf/operational/odl-interface-meta:bridge-ref-info/

http://10.10.10.77:8181/restconf/config/opendaylight-inventory:nodes

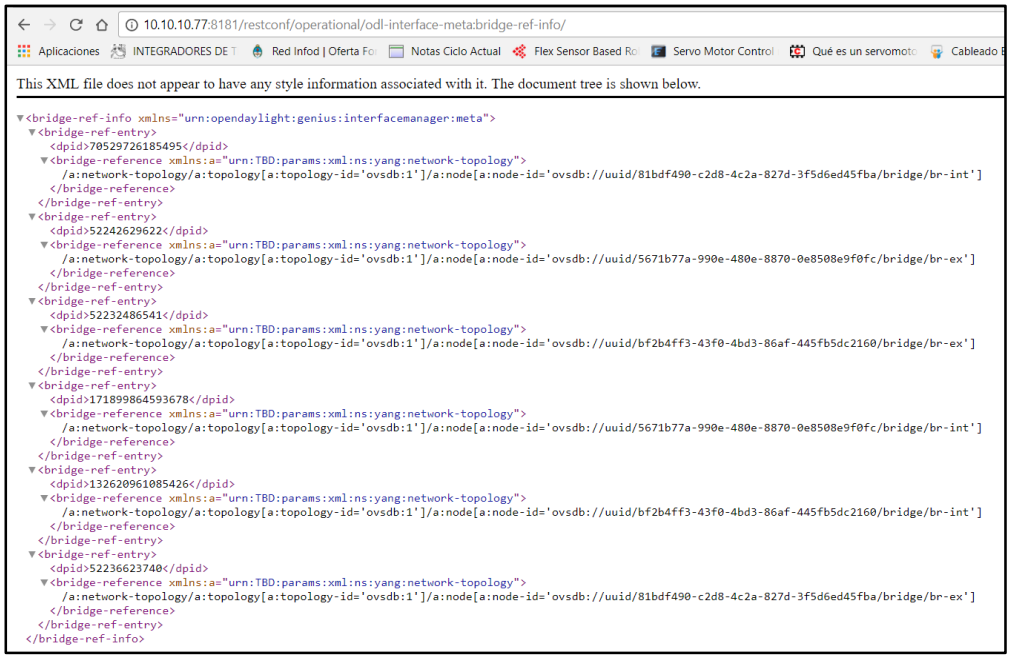


Figura 119. Consulta al datastore de OpenDaylight mediante http.

4.3.2. CAPTURA DE TRÁFICO DE SESIÓN OPENFLOW

Para este punto el controlador OpenDaylight se encuentra gestionando y administrando tres sesiones OpenFlow y varios elementos virtuales a base de reglas Openflow. Para verificar el correcto funcionamiento del controlador y de las sesiones establecidas con cada nodo se realizan capturas de paquetes en cada canal de comunicación con tráfico Openflow.

Mediante el uso de filtros en Wireshark es posible discriminar el tráfico para ver únicamente el relacionado con los nodos. Para la prueba y verificación se captura tráfico de las sesiones y se rastrea los mensajes de la sesión usando el complemento TCP Stream[16] usando el filtro

[16] Se conocen mejor como flujos de datos, en análisis de paquetes se usa los términos TCP Stream y UDP Stream.

openflow_v4 que muestra únicamente los mensajes de Openflow 1.3. En la Figura 120 se muestran los streams Openflow de las tres sesiones correctamente establecidas con el controlador de red.

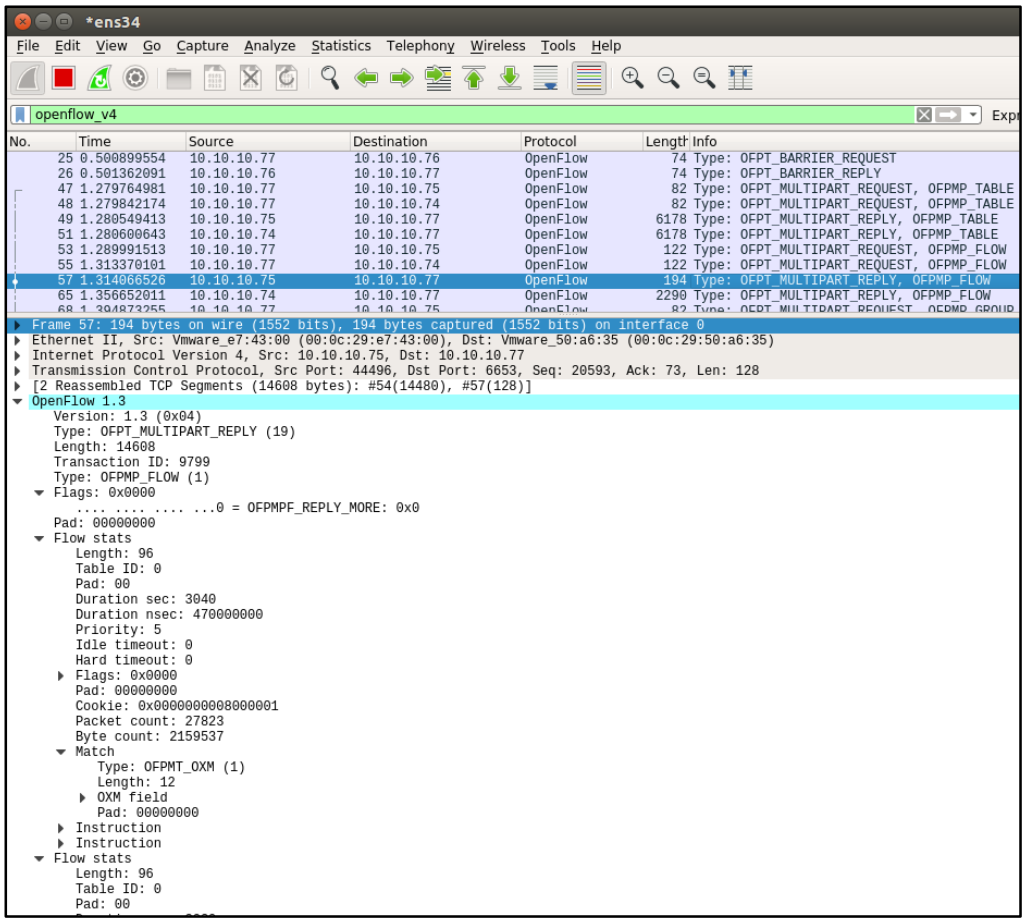


Figura 120. Captura de tráfico de sesiones Openflow en OpenDaylight.

4.4. TRÁFICO VIRTUALIZADO

El tráfico de las redes virtuales es muy parecido al de las redes físicas. En la mayoría de los casos ambas difieren únicamente en el nombre de los dispositivos. Por ejemplo, las tarjetas de red denominadas NIC's son ahora llamadas vNIC's, los switches son ahora vSwitches y funcionan de manera similar a las versiones físicas con la particularidad de no permitir su configuración de manera común.

Lo mismo sucede con la provisión de servicios ya que al encontrarse en una plataforma que provee virtualización en todos los aspectos incluido el área de redes, lo único que cambia es la forma en la que viaja el trafico entrante y saliente hacia la instancia que provee el servicio usando la red virtualizada conectada a la red externa.

4.4.1. TRÁFICO CON ENCAPSULAMIENTO VXLAN

La comunicación entre los nodos usa el método de tunelizacion para redes overlay[17], que permite la separación lógica de los datos que circulan a través de la infraestructura cloud. Es por esta razón que si se realiza una captura de trafico directamente en la interfaz física del nodo de computo aparecerán paquetes con información ilegible. Para poder analizar un paquete con encapsulamiento es necesario usar un disector o un decodificador de paquetes como el que proporciona Wireshark.

Para visualizar el contenido y los campos de un paquete con encapsulamiento VXLAN es posible usar la herramienta *tshark* sobre la línea de comandos para generar un registro de tráfico que luego puede ser leído con más detenimiento en Wireshark mediante la interfaz gráfica. La instrucción que se usa para generar este registro cuenta con un filtro para el puerto default de VXLAN que es 4789 con la siguiente sintaxis.

tshark -i ens34 -d udp.port==4789,vxlan -w 1.cap

Para poder hacer uso de la instrucción anterior es necesario crear el archive donde se guardará el registro de captura de tráfico. Luego de eso se debe esperar un tiempo prudencial para que se

[17] Es la virtualización de nodos enlazados lógicamente, constituida de una o varias redes subyacentes.

registren los paquetes suficientes para el análisis, y posteriormente se debe extraer el archivo del servidor para leerse en wireshark.

La Figura 121 muestra el contenido de un fichero generado durante la captura de paquetes con encapsulamiento VXLAN. En ella se puede apreciar que los paquetes ICMP ya se encuentran decodificados, esto se debe a que la versión de Wireshark para Windows ya incorpora el disector Openflow. Entre los paquetes capturados se puede ver las direcciones IP de origen y destino de las instancias que se encuentran dentro de OpenStack, así como los puertos origen y destino del túnel VXLAN.

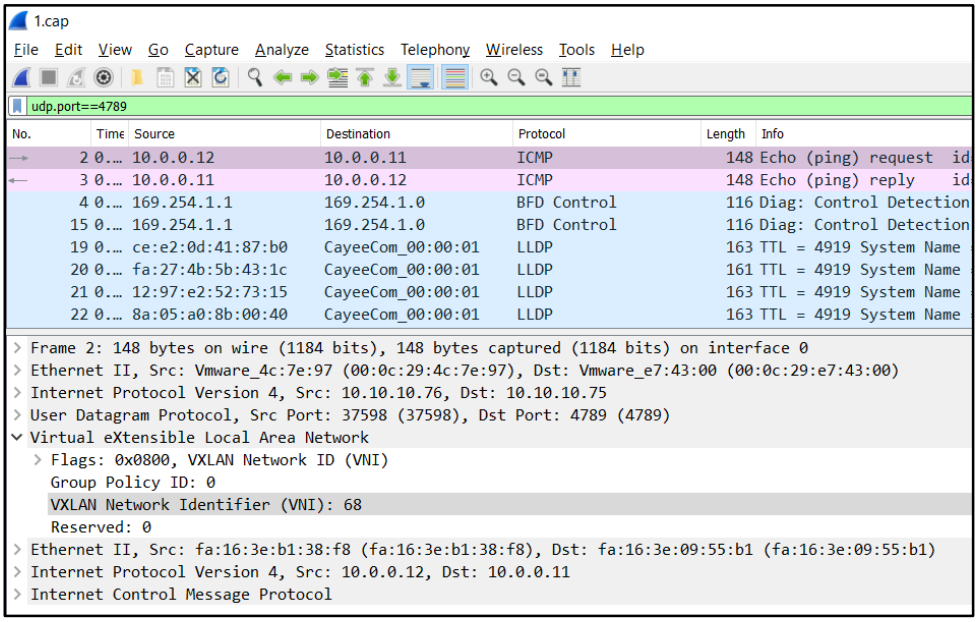


Figura 121. Tráfico VXLAN decodificado en Wireshark.

Para corroborar que la información proporcionada es correcta, se puede consultar el datastore de OpenDaylight para verificar el VNI del túnel que en este caso es 68. Para ello se usa la interfaz API para consultar la sección networks y comparar si el valor es el mismo. Desde la línea de comandos se ejecuta la instrucción para obtener el resultado de la Figura 122.

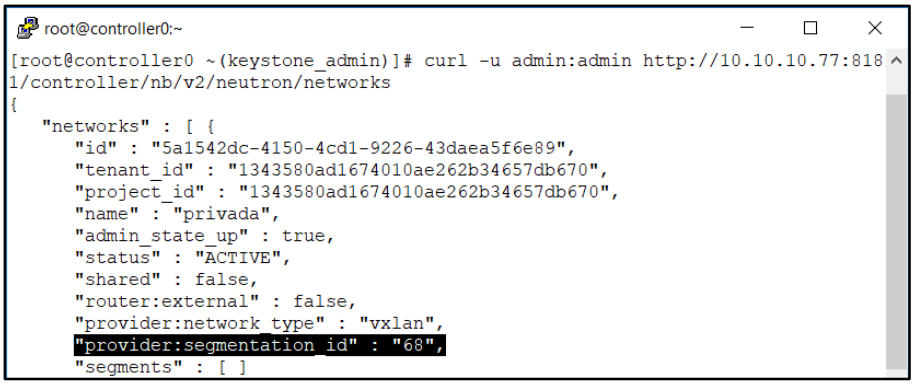


Figura 122. Verificación de VNI VXLAN mediante consulta al datastore de OpenDaylight.

Volviendo a wireshark y desglosando la información del paquete capturado, se puede observar algo muy particular como es la doble cabecera que aparece como Ethernet II. Y entre ellas se encuentra la información del paquete UDP que viaja por el túnel, así como la cabecera de VXLAN que consta de varios campos de extensión, no usados sino con redes con Políticas de Grupo.

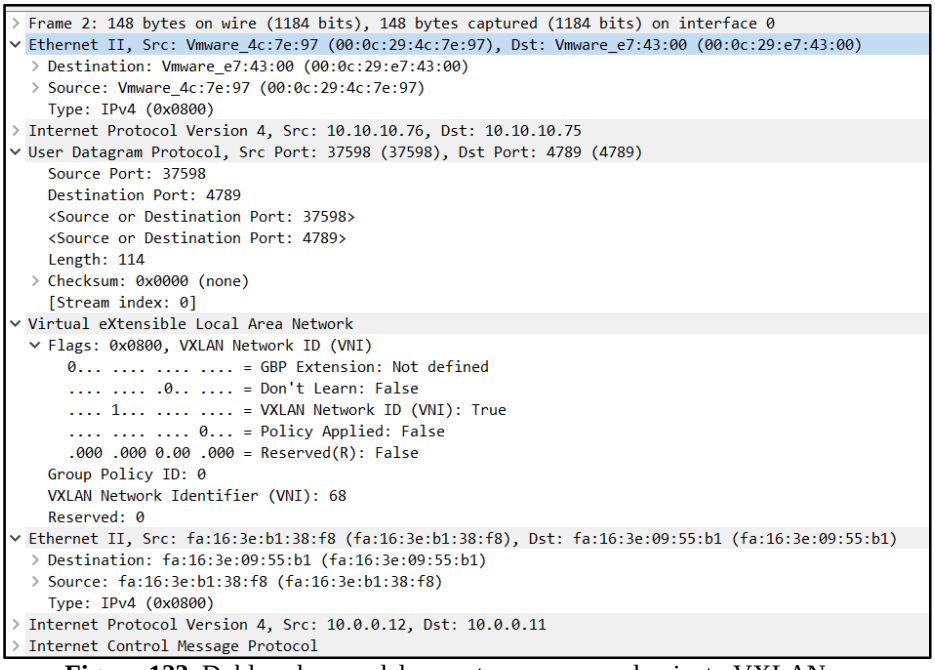


Figura 123. Doble cabecera del paquete con encapsulamiento VXLAN.

4.4.2. VERIFICACIÓN DE TÚNELES EN EL DATASTORE DE OPENDAYLIGHT

En el presente despliegue se usan túneles con encapsulamiento VXLAN en el anillo de comunicación entre nodos. Estos túneles funcionan sobre interfaces independientes del bridge externo y el bridge de integración. El tráfico entre el controlador y los nodos de computo se etiquetan con un VNI de proyecto y se encapsulan para viajar sobre esta red de tipo overlay.

Los paquetes se envían a través de un puerto de túnel predefinido en las tablas de configuración de Open vSwitch, en este caso *ens34*. La sintaxis de esta configuración se envía a OpenDaylight a través de la interfaz OVSDB mediante el parámetro *local_ip*. Este túnel hace uso del puerto de comunicaciones 4789 de manera predeterminada, creando en algunos casos una interfaz especial con el nombre *vxlan_sys_4789.* La Figura 124 describe la interfaz creada por el túnel en el nodo principal.

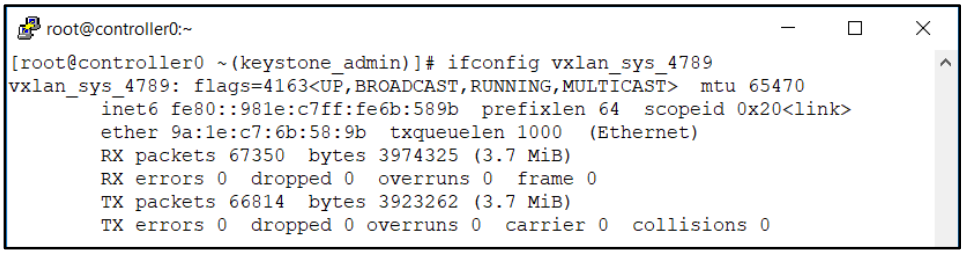


Figura 124. Interfaz de tunelizacion vxlan_sys_4789

De la misma manera como existe la interfaz virtual en cada nodo se registra una entrada en el datastore de OpenDaylight por cada túnel que se genera entre los nodos. La consola de Karaf permite consultar los puertos generados mediante el módulo de integración con Neutron, así como los túneles VXLAN creados a partir de las configuraciones de OpenStack. La Figura 125 muestra los puertos y túneles registrados en OpenDaylight.

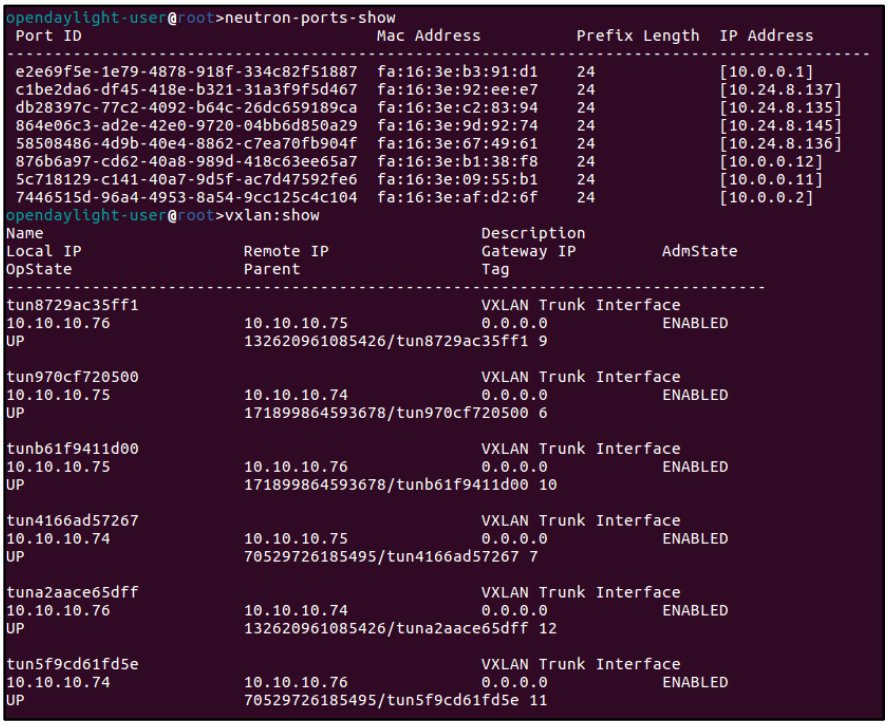


Figura 125. Puertos y túneles registrados en el datastore de OpenDaylight.

El administrador de túneles internos crea y mantiene la malla de túneles de tipo VXLAN o GRE, entre los switches Openflow que forman una red de transporte superpuesta. ITM tiene dos modos de funcionamiento, el primero crea los túneles bajo de manda es decir de manera automática y esta es la opción habilitada por defecto. El segundo modo es manual y permite ingresar túneles pre configurados vía REST_API. Cuando se configura de esta manera es importante establecer correctamente las zonas de transporte ya que de otra manera pueden faltar reglas y la conexión fallará. En la figura 126 se representa mediante DLUX el anillo de túneles creado de manera automática por ITM.

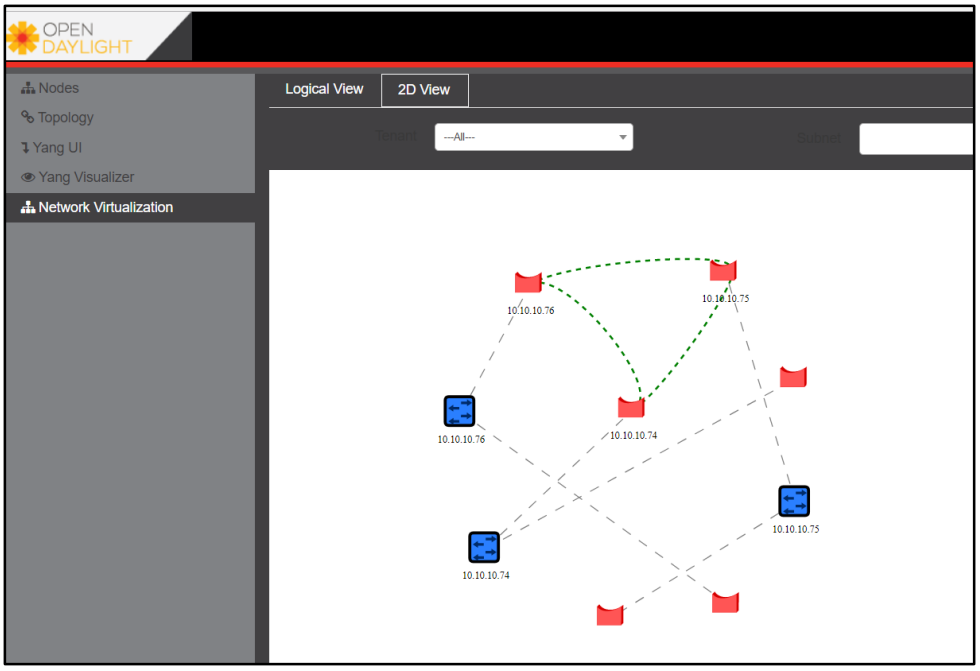


Figura 126. Anillo de túneles VXLAN creados por ITM.

4.4.3. REGLAS OPENFLOW DE LOS TÚNELES

En esta sección se revisan las reglas OpenFlow que se establecen para la comunicación a través de los túneles VXLAN. Para esto se hace referencia a la MAC de la instancia dentro de OpenStack. Se considera *fa:16:3e:09:55:b1* como la dirección física de la máquina virtual y se rastrea los flujos relacionados como se observa en la Figura 127, obteniéndose como resultado dos flujos. El primero se encuentra en la tabla 21 que está relacionada con la FIB y es la encargada de reenvío de capa 3. El segundo flujo se encuentra apuntado en la tabla 51 que se encarga de los filtros de MAC destino. En el campo tun_id se puede apreciar el VNI del túnel en estado hexadecimal. El ANEXO E contiene el pipeline completo de NetVirt y todas las tablas involucradas.

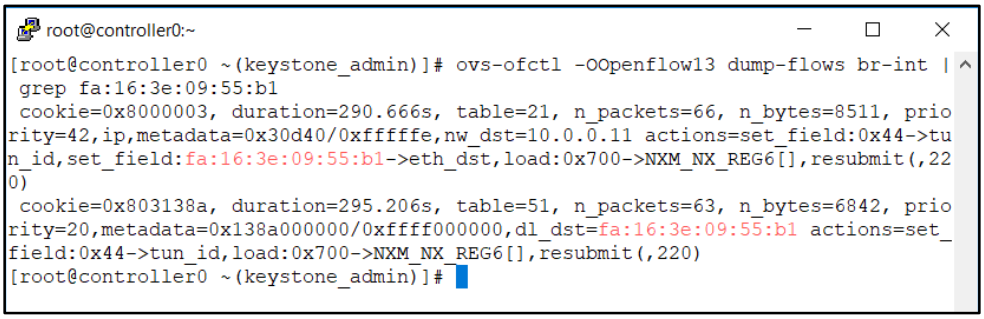


Figura 127. Flujos Openflow de los túneles VXLAN.

4.5. INSTANCIAS OPENFLOW

Una instancia Openflow es la definición de cualquier dispositivo lógico creado a partir de reglas OpenFlow o que use la especificación para establecer sus comunicaciones. En muchos casos dejan de ser los dispositivos en sí y pasan a ser líneas de programación que son capaces de generar el comportamiento de un servicio como es el caso de firewalls, IDS y los mismos switch Openflow.

Durante el despliegue de la red se considera la creación de un router que interconecta el bridge de integración con la red externa a la plataforma cloud que está usando NetVirt. En un inicio estos dispositivos se generan como namespaces de linux, pero cuando el switch virtual para a manos del controlador SDN el router se implementa cien por ciento a base de reglas Openflow lo que le convierte en una instancia Openflow. Esto se puede verificar realizando la consulta de namespaces en el nodo y de dispositivos activos en el datastore de OpenDaylight que es el único lugar donde se ve reflejado.

Figura 128. Instancia Openflow reflejada únicamente en el datastore.

4.6. ESTADÍSTICAS DE FUNCIONAMIENTO DEL CLÚSTER

Mediante el uso de la herramienta *free* mostramos la cantidad de memoria que se está utilizando, en este caso los valores no exceden el consumo reglamentario ya que el clúster se encuentra optimizado para distribuir la carga entre los nodos que posean recursos disponibles. La Figura 129 muestra una comparación paralela del consumo de los servidores miembros del clúster.

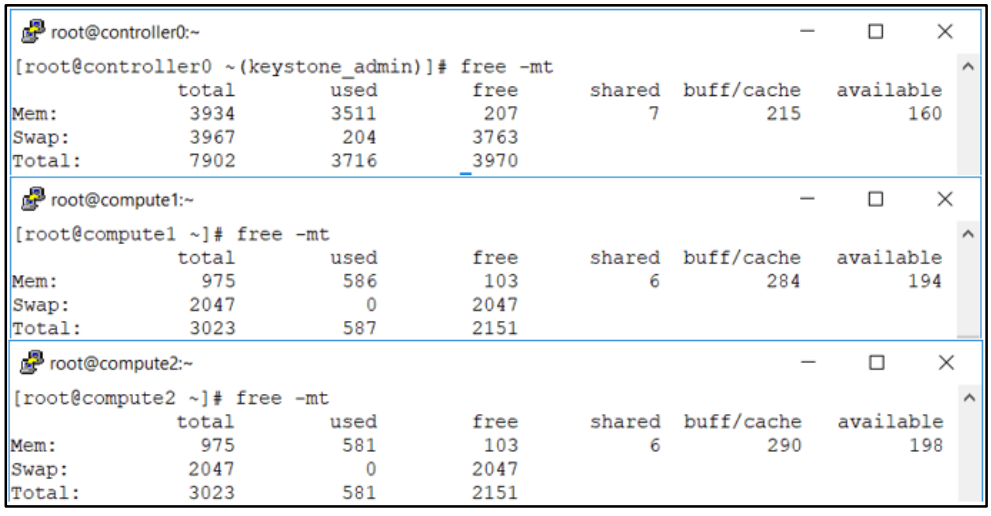


Figura 129. Consumo de memoria de los miembros del clúster.

La herramienta *top* provee una vista dinámica en tiempo real del sistema. Además de mostrar un resumen del uso de recursos del sistema, muestra la lista de tareas que se están siendo manejadas por el kernel de linux. En la Figura 130 se muestra el uso de recursos en los nodos de computo que mantienen funcionando las instancias de prueba.

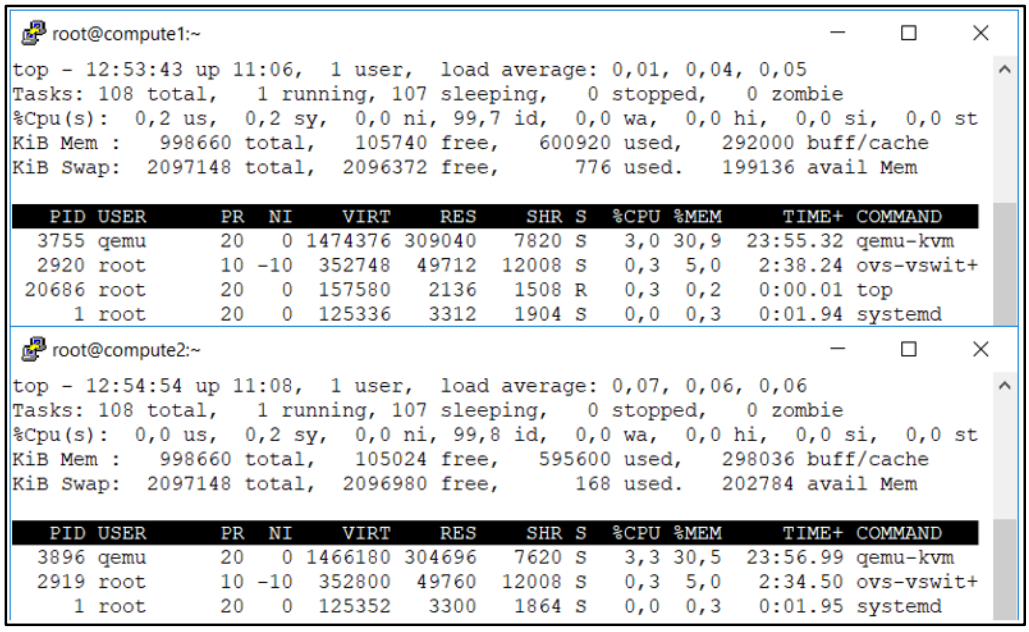


Figura 130. Uso de recursos en los nodos de virtualización.

4.7. STRESS Y CARGA PROMEDIO DE LA INFRAESTRUCTURA FÍSICA

Para obtener un aproximado de la carga promedio de la infraestructura física se somete todos los nodos de virtualización a una prueba de stress mediante la herramienta del mismo nombre. Para ello usamos el comando *uptime* para verificar la carga promedio antes de ejecutar la prueba y luego volvemos a ingresarlo para evidenciar el cambio que sufren los valores de carga promedio. En la Figura 131 se muestra el resultado de las pruebas de stress en los nodos de computo.

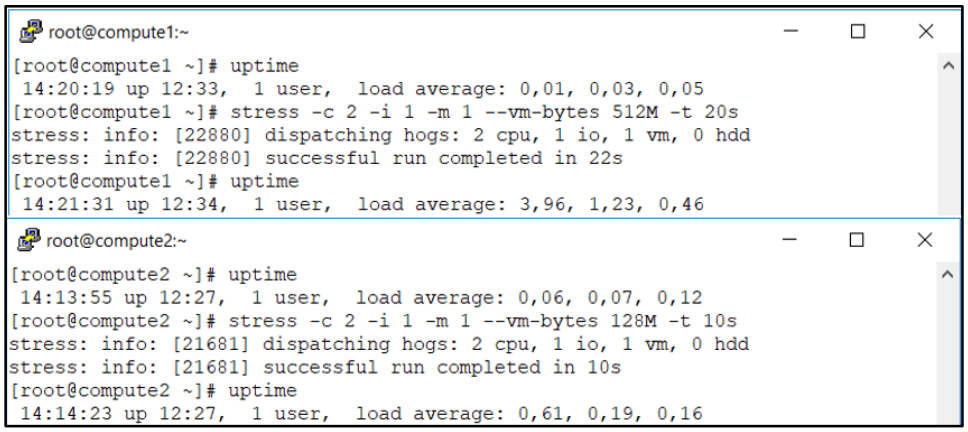


Figura 131. Carga promedio luego de la prueba de stress en los nodos de computo.

De los resultados obtenidos en la prueba anterior se puede decir que, la estabilidad de los nodos es bastante alta y que mientras no se excedan el uso de recursos que se encuentran a disposición, el funcionamiento será normal. Para el caso del controlador de red podemos hacer uso del monitor del sistema en el que se mostrara el histórico de consumo de los recursos. En la Figura 132 se puede observar el resumen grafico del consumo de recursos de OpenDaylight en tiempo real.

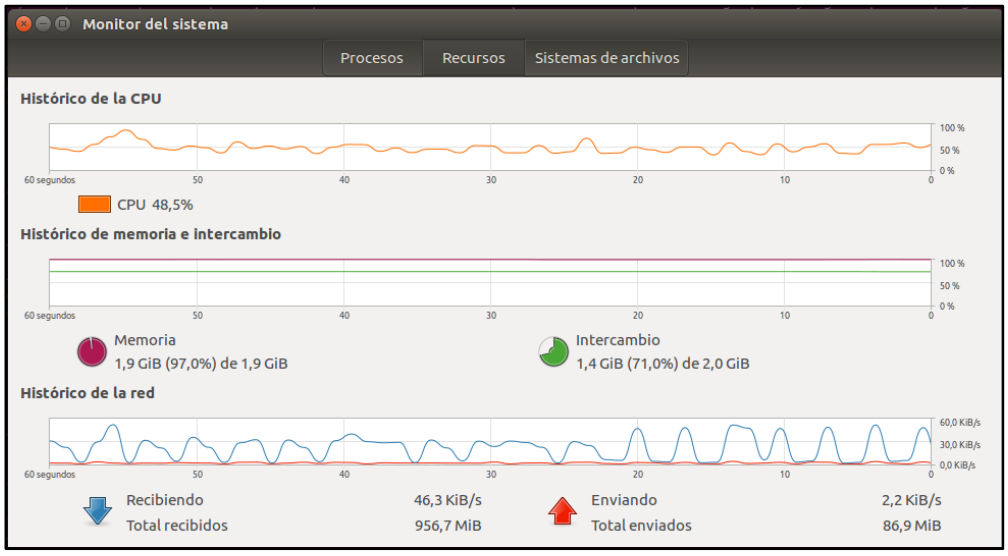


Figura 132. Histórico de consumo en el servidor OpenDaylight.

4.8. ACCESO EXTERNO A LOS RECURSOS VIRTUALES

Como es evidente una de las pruebas restantes es comprobar el funcionamiento de los dispositivos virtualizados a través del acceso desde la red física. Para esto es necesario usar el concepto de IP flotante que introduce OpenStack, pero haciendo uso de los recursos almacenados en el datastore de OpenDaylight. En estas circunstancias el gestor de infraestructura únicamente hace de interprete permitiendo que el administrador manipule las direcciones de manera normal como si del agente de capa 3 de Neutron se tratase y de esta manera se genere una dirección de acceso externo para la instancia. La Figura muestra el proceso de creación de la IP flotante.

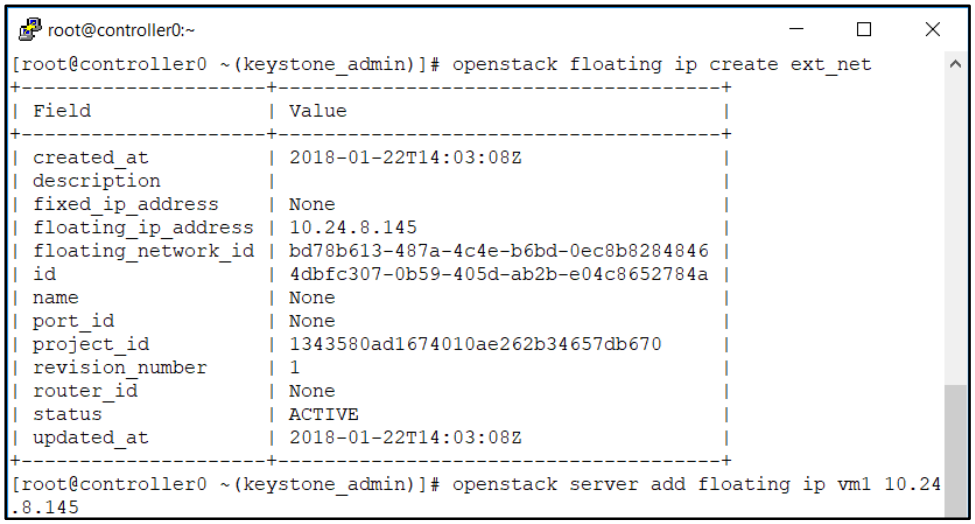


Figura 133. Creación de IP flotante usando la red Openflow.

Para crear dicha IP se puede hacer uso de la interfaz gráfica de usuario de OpenStack o la línea de comandos de Neutron. Para efecto de las pruebas de funcionamiento se usa la línea de comandos en el servidor principal y se genera una dirección a partir del pool de la DMZ de la Universidad Técnica del Norte. Una vez que se crea la dirección se asigna a la instancia y se intenta el acceso mediante SSH desde un host remoto. En la Figura se indica el establecimiento de una sesión SSH desde el host remoto hacia la instancia en OpenStack.

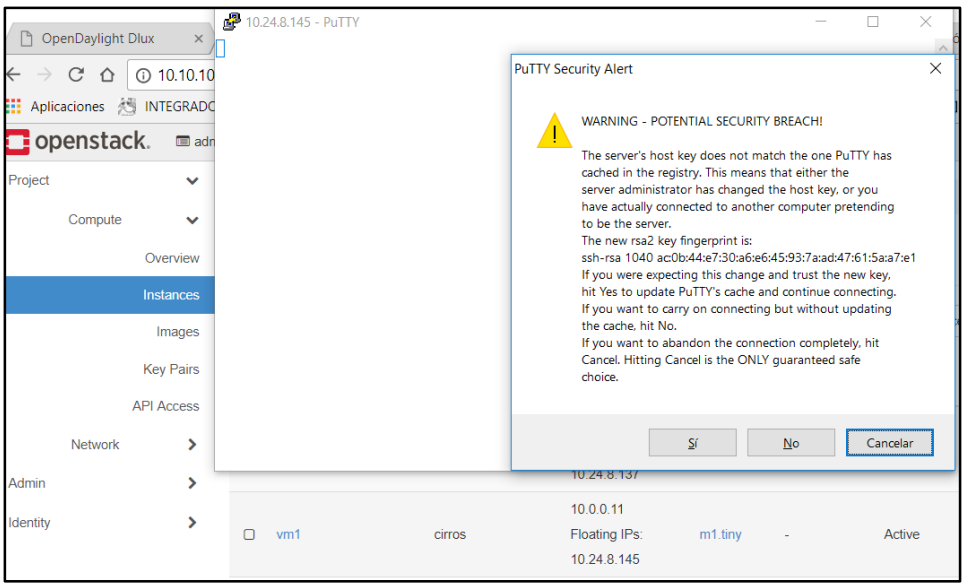


Figura 134. Acceso desde el exterior a instancia en OpenStack.

Para corroborar que el acceso se realiza a través de la SDN se revisan las tablas de flujo del nodo en el que se encuentra alojada la instancia y se inspecciona las reglas de flujo relacionadas con las direcciones virtual y externa. Es posible que gran cantidad de reglas se encuentren instaladas en el switch virtual, pero esto es normal ya que NetVirt se encarga de crear las entradas adecuadas para establecer el Data Path de Openflow. En la Figura se muestran las reglas de flujo de Open vSwitch que se encuentran en el nodo compute1.

Figura 135. Reglas Openflow relacionadas con la IP flotante.

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

- En un inicio varios elementos del Centro de Datos FICA no se encontraban gestionados. Mediante la reestructuración y asignación de roles a los servidores miembros de la infraestructura cloud, se logró mejorar el orden y la administración de los recursos que se encontraban repartidos en tres plataformas diferentes.

- La implementación de la red definida por software para el Centro de Datos FICA se realizó a base de los elementos y herramientas de virtualización de código abierto proporcionados por OpenStack y Opendaylight de manera satisfactoria, habiéndose realizado las pruebas de convergencia necesarias para ratificar la validez del despliegue, así como como la culminación del proyecto.

- Entre las alternativas posibles para el controlador de red, se sometió a evaluación y pruebas previas únicamente a los proyectos que durante el desarrollo de este despliegue se encuentran activos y mantienen soporte de sus creadores. La matriz de selección dio como resultado que OpenDaylight es el proyecto actualmente más robusto y compatible con SDN y los módulos de la plataforma OpenStack.

- La implantación de una red definida por software presenta ciertos retos y requerimientos específicos que las redes comunes. OpenDaylight es una solución flexible y escalable ya que se encarga de realizar las tareas de manera automatizada en ambientes centralizados y descentralizados, tomando control de los switch virtuales permitiendo el despliegue de servicios de manera más rápida.

- OpenStack presenta grandes ventajas frente a otros gestores de infraestructura ya que su naturaleza modular le permite integrarse con software de terceros, como en el caso del módulo de red Neutron_ML2 que facilita el acoplamiento con el controlador de red OpenDaylight mediante el driver networking_odl. A la vez el uso del mecanismo de despliegue de NetVirt permitió reducir la cantidad de puntos de falla que pudieron haber existido al realizar la implementación de manera manual, ya que la cantidad de elementos y reglas necesarias para poner en funcionamiento el entorno es bastante elevada. Gracias a esto la creación de las rutas para el plano de datos se efectuó de manera más sencilla.

- Durante la ejecución del despliegue se presentaron una serie de dificultades relacionadas con términos y temas de Openflow que se solucionaron mediante el uso de las herramientas puestas a disposición por los desarrolladores, como repositorios y fuentes documentales de proyectos anteriores a NetVirt como OVSDB.

- Después de haberse efectuado la implementación del controlador de red correctamente, se pudo comprender los alcances de la integración de plataformas, permitiendo proporcionar servicios más robustos, más fiables y con un índice de escalabilidad mucho mayor al prestado usando únicamente gestores de servicios de manera individual.

- El pipeline de NetVirt es bastante amplio por lo que puede volverse complicado de entender en un inicio. Por esta razón durante el despliegue se seleccionó cuidadosamente las funciones que se instalarían en el controlador, de esta manera cuando se generaron las reglas de flujo la lectura y el rastreo de los datapaths no fue muy complejo.

- Muchos de los usos que se puede dar a la API_REST de OpenDaylight ya no son tan complicados, ya que mediante Yangman es posible realizar consultas e incluso configuraciones

en el datastore del controlador. Considerando la fragilidad actual de las pilas de datos que almacena el controlador puede llegar a ser critica la manipulación de esta información sin el conocimiento necesario, en especial en ambientes de producción.

- Los dispositivos totalmente definidos por software presentan un nivel de sensibilidad bastante alto por lo que fue necesario analizar el conjunto de reglas que los constituyen para solucionar problemas de conectividad en caso de reglas faltantes o bloqueos no intencionales generados por flujos inconsistentes. De la misma manera se evidenció que los dispositivos totalmente OpenFlow no son alcanzables mediante solicitudes de ping. Se comprobó que al ser cien por ciento Openflow no manejan los paquetes ICMP como tal. Para la realización de las pruebas fue necesario usar la herramienta arp-ping a la cual los dispositivos si responden ya que en sus tablas existe un ARP Responser.

- El tráfico encapsulado con VXLAN sobre redes virtuales representa una ventaja ante los dispositivos físicos competidores ya que estos últimos necesitan de algún elemento que gestione el des encapsulamiento de la red para inyectarlo en una red virtual diferente o a la red física. Es por eso que el uso de las SDN se va convirtiendo cada vez más en un acierto antes que un reto.

- Las redes superpuestas con encapsulamiento VXLAN permiten crear un gran número de pequeños segmentos aislados de red más allá de los 4000, lo que hace que la arquitectura multi tenant se vuelva más escalable. De esta manera durante la etapa de despliegue se tuvo la facilidad de crear y destruir redes virtuales mientras se conseguía que el mecanismo NetVirt funcione adecuadamente.

RECOMENDACIONES

- Incursionar en el despliegue de redes definidas por software representa un sinnúmero de requerimiento tanto de infraestructura como de conocimientos técnicos, por lo que se recomienda mantener constante el desarrollo de proyectos relacionados con temas de virtualización y software para redes.

- Durante el proceso de despliegue de la red se presentaron una serie de inconvenientes críticos como la suspensión del suministro eléctrico en el centro de datos, lo cual entorpeció las tareas de implementación. Por esta razón se sugiere mantener siempre un script actualizado de las configuraciones llevadas a cabo en caso de ser necesaria una reinstalación de los componentes.

- Antes de realizar la selección de cualquier software relacionado con virtualización y redes de comunicación es necesario consultar los repositorios cuales son las versiones más estables para evitar incompatibilidades, como el caso de Java 9 con el controlador OpenDaylight lo cual causo un mal funcionamiento del mismo.

- La configuración de equipos de red y servidores requiere que se maneje por lo menos dos lenguajes de programación, para de esta manera tener opciones al momento de ejecutar tareas de integración de plataformas o creación de módulos y aplicaciones para cualquier sistema que se encuentre en producción.

- Al momento de suscitarse algún error o inconveniente en el despliegue es imperativo buscar la causa del error acudiendo a los logs de eventos y servicios antes de realizar modificaciones indebidas en componentes que no están relacionados con el error. Generalmente los errores se generan por la configuración equivocada de algún fichero o en el caso más extremo un bug en el código del programa.

- Al momento de poner en marcha OpenDaylight en su versión más reciente Nitrogen se generaron varios errores por bugs, razón por la cual fue necesario el downgrade a una versión más estable. Se recomienda visitar la página del proyecto para consultar las posibles fallas q pueden suceder antes de poner el software en producción.

- En el presente despliegue se usaron gran cantidad de usuarios y contraseñas para todos los servicios que forman parte del gestor de infraestructura y de los servicios que son parte del clúster. Por esta razón se debe mantener una tabla organizada con los accesos más críticos, para que puedan ser localizados de manera sencilla en caso de requerir algún cambio y evitando perdida de datos.

- Cuando se trabaja con software de virtualización es recomendable hacer uso de la característica de instantáneas, lo que permitirá volver a un estado anterior en caso de que algún servicio, fichero o instancia se corrompa por mala manipulación o actualizaciones con errores. De esta manera se mantiene un respaldo de los datos de manera paulatina.

- Para futuros desarrollos o despliegues sobre ambientes reales o de prueba, es importante incorporar los servicios que no se han tomado en cuenta en el presente despliegue por ser temas más amplios. De esta manera se podrá tener una visión más profunda del alcance del controlador OpenDaylight, del gestor de infraestructura OpenStack y de las mismas redes definidas por software.

REFERENCIA BIBLIOGRÁFICA

Abdulkhareem, S. (Agosto de 2011). *DRBD – Heartbeat (Active/Passive High Availability Cluster)*. Obtenido de https://syslint.com/blog/tutorial/drbd-–-heartbeat-activepassive-high-availability-cluster/

Apache Software Foundation. (2017). *Apache CloudStack: About*. Obtenido de https://cloudstack.apache.org/about.html

Azodolmolky, S. (2013). *Software Defined Networking with Openflow.* Birmingham: Packt Publishing.

Bermeo, J., Sanchez, M., Maldonado, J., & Carvallo, J. (2005). *Modelos de Calidad de Software en la Práctica: Mejorando su Construcción con el Soporte de Modelos Conceptuales.* Obtenido de http://repositorio.cedia.org.ec/bitstream/123456789/1001/1/Mapeo_MC_MC.pdf

CISCO. (2015). *Event-Based Software-Defined Networking: Build a Secure Science DMZ.* California.

CISCO. (Enero de 2016). *Cisco Application Virtual Switch and VMware vSphere Distributed Switch Failover Convergence in Cisco Application Centric Infrastructure*. Obtenido de https://www.cisco.com/c/en/us/products/collateral/switches/application-virtual-switch/white-paper-c11-736554.html

CISCO. (2017). *Cisco Open SDN Controller*. Obtenido de https://www.cisco.com/c/en/us/products/cloud-systems-management/open-sdn-controller/index.html

CiscoDevNet. (2017). *OpenDaylight YANGMAN*. Obtenido de https://github.com/CiscoDevNet/yangman

de Pozuelo, R. (2013). *Networking and Internet Tecnologies*. Obtenido de http://blogs.salleurl.edu/networking-and-internet-technologies/foro-tecnologico-aslan-avanzando-hacia-la-sdn/

Donovan, J., & Prabhu, K. (2017). *Building the Network of the Future.* Boca Raton: CRC Press.

Emergya. (Enero de 2016). *Curso OpenStack.* Obtenido de http://iesgn.github.io/emergya/curso/u8/openvswitch

Erickson, D. (2013). *Beacon Releases*. Obtenido de https://openflow.stanford.edu/display/Beacon/Releases

Fernandes, L. (Julio de 2017). *OpenFlow 1.3 Software Switch*. Obtenido de https://github.com/CPqD/ofsoftswitch13/blob/master/README.md

Floodlight Project. (2017). *Floodlight*. Obtenido de Floodlight Is an Open SDN Controller: http://www.projectfloodlight.org/floodlight/

FUJITSU. (2017). *Storage Cluster - to ensure business continuity*. Obtenido de http://www.fujitsu.com/global/products/computing/storage/disk/eternus-dx/feature/failover.html

GitHub Developers. (Octubre de 2012). *NOX Repo*. Obtenido de www.noxrepo.org

GitHub Developers. (Octubre de 2013). *OpenFlow 1.3 for OpenWRT*. Obtenido de https://github.com/CPqD/ofsoftswitch13/wiki/OpenFlow-1.3-for-OpenWRT

Goasguen, S. (2014). *60 Recipes for Apache CloudStack.* O'Reilly.

Google Developers. (Octubre de 2017). *Single node clusters*. Obtenido de https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/single-node-clusters

Goransson, P., Black, C., & Davy, M. (2014). *Software Defined Networks A Comprehensive Approach.* Elsevier.

Hamburguer, V. (2016). *Building VMware Software-Defined Data Centers* . Packt Publishing.

HP Enterprise. (Junio de 2014). *hpe.com.* Obtenido de hpe.com: https://www.hpe.com/h20195/V2/GetPDF.aspx/4AA5-0092ENW.pdf

IMPRESSICO BUSINESS SOLUTIONS. (Noviembre de 2017). *Understanding the Cloud: The SPI Model*. Obtenido de http://www.impressico.com/2015/11/25/understanding-the-cloud-the-spi-model/

Interoute Iberia. (Julio de 2014). *La estrecha relación de SDN, NFV y Cloud Computing*. Obtenido de https://www.interoute.es/blog/estrecha-relacion-sdn-nfv-cloud-computing/

Khedher, O. (2015). *Mastering OpenStack.* Packt Publishing.

Linux Foundation. (2016). *Production Quality, Multilayer Open Virtual Switch*. Obtenido de http://openvswitch.org

Linux Foundation Administrators. (Septiembre de 2017). *What is ONOS?* Obtenido de https://wiki.onosproject.org

LIVERPOOL UNIVERSITY. (Noviembre de 2014). *UNIVERSITY OF LIVERPOOL*. Obtenido de UNIVERSITY OF LIVERPOOL: http://clusterinfo.liv.ac.uk

Microsoft. (Octubre de 2008). *Design for a Failover Cluster with Multiple Services and Applications*. Obtenido de https://technet.microsoft.com/en-us/library/dd197459(v=ws.10).aspx

Microsoft Developer Network. (2003). *Load-Balanced Cluster*. Obtenido de https://msdn.microsoft.com/en-us/library/ff648960.aspx

Microsoft TechNet. (Marzo de 2003). *What Is a Server Cluster?* Obtenido de https://technet.microsoft.com/en-us/library/cc785197(v=ws.10).aspx

Mininet Project. (2017). *Mininet Documentation.* Obtenido de https://github.com/mininet/mininet/wiki/Documentation

Morreale, P. A., & Anderson, J. M. (2015). *Software Defined Networking Design and Deployment.* CRC Press.

Nadeau, T. D., & Gray, K. (2013). *SDN Software Defined Networking.* O'REILLY.

Noll, M. (Julio de 2011). *Applied Research. Big Data. Distributed Systems. Open Source.* Obtenido de http://www.michael-noll.com: http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/

ONOS. (Septiembre de 2016). *ONOS System Components*. Obtenido de https://wiki.onosproject.org/display/ONOS/System+Components

Open Networking Fundation. (2017). *Software-Defined Networking (SDN) Definition.* Obtenido de https://www.opennetworking.org/sdn-definition/

OpenDaylight. (2017). *Platform Overview*. Obtenido de https://www.opendaylight.org/what-we-do/odl-platform-overview

OpenDaylight Project. (Marzo de 2013). *OpenDaylight Controller:Architectural Framework*. Obtenido de https://wiki.opendaylight.org/view/OpenDaylight_Controller:Architectural_Framework

OpenDaylight Project. (Febrero de 2016). Opendaylight Installation Guide.

OpenFlow. (Diciembre de 2009). *OpenFlow Switch Specification 1.0.0*. Obtenido de http://archive.openflow.org/documents/openflow-spec-v1.0.0.pdf

OpenFlow. (Febrero de 2011). *OpenFlow Switch Specification 1.1.0.* Obtenido de http://archive.openflow.org/documents/openflow-spec-v1.1.0.pdf

OpenFlow. (Diciembre de 2011). *OpenFlow Switch Specification 1.2.* Obtenido de https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2014/10/openflow-spec-v1.2.pdf

OpenStack. (Noviembre de 2016). *Overview and components*. Obtenido de https://docs.openstack.org/liberty/networking-guide/intro-os-networking-overview.html

OpenStack. (Diciembre de 2017). *Networking architecture*. Obtenido de https://docs.openstack.org/security-guide/networking/architecture.html

ORACLE. (2011). *Guía de administración del servidor Oracle VM para SPARC 2.1*. Obtenido de https://docs.oracle.com/cd/E24621_01/html/E23598/virtualnetworkdevice.html

PLEXXi. (2017). *PLEXXI CONTROL*. Obtenido de http://www.plexxi.com/plexxi-control/

Project Floodlight. (2017). *Indigo Virtual Switch*. Obtenido de http://www.projectfloodlight.org/indigo-virtual-switch/

Red Hat, Inc. (2007). *Cluster Basics*. Obtenido de https://www.centos.org/docs/5/html/Cluster_Suite_Overview/s1-clstr-basics-CSO.html

RedHat Customer Portal. (2017). *Deploying OpenStack using PackStack.* Obtenido de https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform/2/html/Getting_Started_Guide/part-Deploying_OS_using_PackStack.html

SDxCentral. (2017). *2017 Network Virtualization Report SDN Controllers, Cloud Networking and More.* Obtenido de https://www.sdxcentral.com/reports/network-virtualization-sdn-controllers-download-2017/SDxCentral-Network-Virtualization-Report-2017-Rev-A.pdf

SDxCentral. (2017). *SDXCentral*. Obtenido de www.sdxcentral.com: https://www.sdxcentral.com/wp-content/uploads/2014/04/OpenFlow-Controller-Protocol.png

SDxCentral. (2017). *What is Ryu Controller?* Obtenido de https://www.sdxcentral.com/sdn/definitions/sdn-controllers/open-source-sdn-controllers/what-is-ryu-controller/

Subramanian, S., & Voruganti, S. (2016). *Software-Defined Networking (SDN) whit OpenStack.* Packt Publishing.

The Open Networking Foundation. (2013). *OpenFlow Switch Specification 1.4.0.* Obtenido de https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf

Trema. (2017). *Trema Documentation*. Obtenido de http://www.rubydoc.info/github/trema/trema/master/frames

VMware. (2013). *The VMware NSX Network Virtualization Platform.* Palo Alto.

VMware. (2017). *Qué es la virtualización*. Obtenido de http://www.vmware.com/es/solutions/virtualization.html

Xining, H. (Julio de 2016). *Análisis de la tabla de flujo de Netvirt*. Obtenido de http://www.sdnlab.com/17506.html

GLOSARIO DE TÉRMINOS

ACL (Access Control List): Concepto de seguridad informática usado para fomentar la separación de privilegios.

API (Application Programming Interface): Conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

ASIC: Circuito Integrado para Aplicaciones Específicas.

Backbone: Línea de transmisión más grande que transporta los datos recogidos de líneas más pequeñas.

BigData: Término que describe el gran volumen de datos.

Bridging: Creación de un canal de comunicación es genérico a base de dos dispositivos físicamente separados.

CLI: Interfaz de línea de comandos.

Cloud Computing: Servicios informáticos o de computo en la Nube.

Clúster: Grupo de ordenadores unidos entre sí normalmente por una red de alta velocidad y que se comportan como si fuesen una única computadora.

Controlador: Servidor al que se le ha otorgado el rol de manejar de manera centralizada ciertos elementos de una organización.

CP: Espacio donde se ejecutan los procesos que involucran exclusivamente ordenes entre elementos de red.

CSP: Proveedor de servicios que transporta información electrónicamente.

Data Center: Espacio donde se concentran los recursos necesarios para el procesamiento de la información de una organización.

DevOps: Práctica de ingeniería de software que tiene como objetivo unificar el desarrollo de software (Dev) y la operación del software (Ops).

DHCP: Protocolo cliente-servidor que proporciona automáticamente un host de protocolo Internet (IP) con su dirección IP y otra información de configuración relacionados.

DiffServ: Arquitectura que proporciona un método que intenta garantizar la calidad de servicio en redes de gran tamaño, como puede ser Internet.

EMS: Consiste en sistemas y aplicaciones para gestionar elementos de red en la capa de gestión de elementos de red.

Escalabilidad: Habilidad de reaccionar y adaptarse sin perder calidad.

FIB: Base de datos con información de las tablas de reenvío.

Framework: Plataforma o entorno de trabajo enfocado a una problemática en particular.

HA: Protocolo de diseño del sistema y su implementación asociada que asegura un cierto grado absoluto de continuidad operacional.

HAL: Elemento del sistema operativo que funciona como una interfaz entre el software y el hardware del sistema.

HASH: Algoritmos que consiguen crear a partir de una entrada una salida alfanumérica de longitud normalmente fija que representa un resumen.

Hypervisor: Plataforma que permite aplicar diversas técnicas de control de virtualización para utilizar, al mismo tiempo, diferentes sistemas operativos.

IaaS: Acrónimo de Infrastructure as a Service, es una forma de computación en la nube donde se ofrecen a sus clientes recursos, físicos y virtuales.

Instancia: Unidad operativa que posee características heredadas del hardware donde se ejecuta.

IQMC: Método que proporciona las guías y técnicas para identificar los factores de calidad apropiados de los componentes de software.

ITM: Elemento que crea y mantiene malla de túneles entre los switches Openflow que forman una red de transporte superpuesta.

JVM: Máquina virtual de proceso nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial.

LIB: Base de datos existente en una red funcional a base de etiquetas.

Metadatos: Grupo de datos que describen el contenido informativo de un objeto al que se denomina recurso.

Multicast: Envío de la información en múltiples redes a múltiples destinos simultáneamente.

NFV: Marco de referencia general relacionado con la arquitectura de redes, orientado a virtualizar diferentes elementos dentro de las mismas.

NGN: Infraestructura que permite la confluencia de los nuevos servicios multimedia integrados.

Nodo: Punto de conexión de varios elementos de hardware o software.

NOS: Software que permite la interconexión de ordenadores para tener el poder de acceder a los servicios y recursos, hardware y software, creando redes de computadoras.

ONOS: Proyecto desarrollado para crear un sistema operativo de red definido por software (SDN) para los proveedores de servicios de comunicaciones.

Open vSwitch: Software de código abierto, diseñado para ser utilizado como un switch virtual en entornos de servidores virtualizados.

OpenFlow: Protocolo que permite a un servidor decirles a los conmutadores de red adónde enviar paquetes.

Orquestación: Gestión automática o semiautomática de nuestra infraestructura y servicios gracias a alguna API o aplicación.

OSS: Sistemas de información empleados por las empresas operadoras de telecomunicaciones.

Overlay: Red virtual de nodos enlazados lógicamente, que está construida sobre una o más redes subyacentes.

OXM: Función que permite importar/exportar datos de un objeto por medio del mapeo de sus propiedades.

PaaS: Acrónimo de Platform as a Service, es un concepto de computación en la nube mediante la cual los usuarios pueden desarrollar, ejecutar y administrar aplicaciones.

Paralelismo: Ejecución de tareas al mismo tiempo.

PBB: Estándar que adapta la tecnología Ethernet a las redes de transporte de clase de operador.

PoC: Implementación, a menudo resumida o incompleta, de un método o de una idea, realizada con el propósito de verificar que el concepto o teoría.

QoS: Rendimiento promedio de una red de telefonía o de computadoras, particularmente el rendimiento visto por los usuarios de la red.

SaaS: Acrónimo de Software as a Service, es un modelo de distribución de software por el que terceros desarrolladores ofrecen ciertas aplicaciones a través de Internet.

SDN: Conjunto de técnicas relacionadas con el área de redes computacionales, cuyo objetivo es facilitar la implementación e implantación de servicios de red a base de herramientas de software.

SDSF: Arquitectura orientada a la provisión de servicios con un enfoque estratificado donde las capas se definen en términos de un modelo de servicio en la nube.

SFC: Conjunto de capacidades para definir una lista ordenada de servicios de red.

TE: Metodología para adaptar flujos de tráfico a recursos físicos de la red, de tal forma que exista un equilibrio entre dichos recursos.

Tenant: Segmento aislado que posee ciertos privilegios y características establecidas desde el Proyecto principal.

TLV: Representación de datos, de forma que haya información que pueda tener presencia opcional y longitud variable.

Underlay: Red subyacente con infraestructura tradicional.

Virtualización: Creación a través de software de una versión virtual de algún recurso tecnológico.

Vnet: Dispositivo virtual que está definido en un dominio conectado a un conmutador virtual.

VPN: tecnología de red de computadoras que permite una extensión segura de la red de área local (LAN) sobre una red pública.

YANG: Lenguaje de modelado de datos para la definición de datos enviados a través del protocolo de configuración de red NETCONF.

ANEXOS

ANEXO A-1: NORMA ISO/IEC/IEEE 29148

INTERNATIONAL STANDARD

ISO/IEC/ IEEE 29148

First edition
2011-12-01

Systems and software engineering — Life cycle processes — Requirements engineering

Ingénierie des systèmes et du logiciel — Processus du cycle de vie — Ingénierie des exigences

Reference number
ISO/IEC/IEEE 29148:2011(E)

ANEXO A-2: NORMA ISO/IEC/IEEE 2504n

Instituto Ecuatoriano de Normalización

Quito – Ecuador

NORMA
TÉCNICA
ECUATORIANA

NTE INEN-ISO/IEC 25040
Primera edición
2014-03

SISTEMAS E INGENIERÍA DE SOFTWARE – REQUERIMIENTOS Y EVALUACIÓN DE SISTEMAS Y CALIDAD DE SOFTWARE (SQuaRE) – PROCESO DE EVALUACIÓN (IEC 25040:2011, IDT)

SYSTEMS AND SOFTWARE ENGINEERING – SYSTEMS AND SOFTWARE QUALITY REQUIREMENTES AND EVALUATION (SQuaRE) – EVALUATION PROCESS

Correspondencia:

Esta Norma Técnica Ecuatoriana es una traducción idéntica de la Norma Internacional IEC 25040: 2011

DESCRIPTORES: Software
ICS: 35.080

65
Páginas

ANEXO B: MATRICES DE EVALUACIÓN DE CALIDAD IQMC

Matriz de evaluación de calidad para la característica: Funcionalidad

	Características /Subcaracterísticas	Métrica	OpenStack	OpenNebula	Eucalyptus
	FUNCIONALIDAD				
	ADECUACIÓN				
	Adecuación funcional				
1	Administración de virtualización de escritorios	Si= 1/No= 0	1	1	1
2	Aprovisionamiento almacenamiento	Si= 1/No= 0	1	1	1
3	Recursos a proveer	Si= 1/No= 0	1	1	1
	EXACTITUD				
	Precisión de lo esperado				
1	Resultado de pruebas	Si= 1/No= 0	1	1	1
2	Pruebas de terceros	Si= 1/No= 0	1	1	1
	Integridad de Datos				
1	Administrar historial de cambios	Si= 1/No= 0	1	0	1
2	Mecanismos de control de acceso	Si= 1/No= 0	1	1	1
	INTEROPERABILIDAD				
	Interoperabilidad con Servicios Web				
1	Integración con componentes AWS	Si= 1/No= 0	1	1	1
	SEGURIDAD DE ACCESO				
	Control de Acceso				
1	Gestionado por la aplicación	Si= 1/No= 0	1	1	1
2	Nivel de parametrización para acceso a la funcionalidad (perfil, grupo, usuario)	Si= 1/No= 0	1	1	1
	Seguridad de Datos				
1	Datos almacenados	Si= 1/No= 0	1	1	1
2	Datos transmitidos	Si= 1/No= 0	1	1	1

Matriz de evaluación de calidad para la característica: Fiabilidad.

	Características /Subcaracterísticas	Métrica	OpenStack	OpenNebula	Eucalyptus
	FIABILIDAD				
	MADUREZ				
	Pruebas Necesarias				
1	Tiempo en el mercado	[0:4]	4	3	2
2	Actualizaciones disponibles	[0:4]	4	3	3
3	Mantiene una base de conocimiento	Si= 1/No= 0	1	1	1
	TOLERANCIA A FALLOS				
	Latencia de Fallos				
1	Trabajar en clúster	Si= 1/No= 0	1	1	1
2	Eventos y transacciones del sistema	Si= 1/No= 0	1	1	1
3	Respaldo y recuperación segura de datos	Si= 1/No= 0	1	1	1
	Capacidad de restauración				
1	Facilidades de backup y recovery del sistema	Si= 1/No= 0	1	1	1
2	Facilidades de backup y recovery de datos	Si= 1/No= 0	1	1	1
	RECUPERABILIDAD				
1	Facilidad de recuperación del historial	Si= 1/No= 0	1	1	1
2	Facilidad en la recuperación de contenidos eliminados	Si= 1/No= 0	1	1	1

Matriz de evaluación de calidad para la característica: Portabilidad.

	Características /Subcaracterísticas	Métrica	OpenStack	OpenNebula	Eucalyptus
	PORTABILIDAD				
	ADAPTABILIDAD				
	Adaptabilidad de Hardware				
1	Soporte para máquinas virtuales Linux	Si= 1/No= 0	1	1	1
2	Soporte para máquinas virtuales Windows	Si= 1/No= 0	1	1	1
	COEXISTENCIA				
	Coexistencia				
1	Coexistencia de base de datos	Si= 1/No= 0	1	1	1
2	Coexistencia con servidores Amazon EC2	Si= 1/No= 0	1	1	1

Matriz de evaluación de calidad para la característica: Mantenibilidad

	Características /Subcaracterísticas	Métrica	OpenStack	OpenNebula	Eucalyptus
	MANTENIBILIDAD				
	CAPACIDAD PARA SER APROBADO				
	Capacidad para ser aprobado				
1	Tiempo de instalación	[0:4]	4	4	3
2	Provee editores de formas y vistas	Si= 1/No= 0	1	1	1
	CAPACIDAD PARA SER ANALIZADO				
1	Mantenimiento fácil	Si= 1/No= 0	1	1	1
2	Manejo de estándares	Si= 1/No= 0	1	1	1
	ESTABILIDAD				
1	Frecuencia de actualizaciones por corrección de errores	[0:4]	3	1	2
2	Frecuencia de nuevas versiones	[0:4]	3	4	2

Matriz de evaluación de calidad para la característica: Eficiencia

	Características /Subcaracterísticas	Métrica	OpenStack	OpenNebula	Eucalyptus
	EFICIENCIA				
	COMPORTAMIENTO EN EL TIEMPO				
	Rendimiento				
1	Actualizaciones del contenido automáticamente	[0:4]	3	1	1
2	Tiempo de respuesta promedio en operaciones	[0:4]	3	2	2
	UTILIZACION DE RECURSOS				
	Recursos de Hardware				
1	Características de hardware	[0:4]	4	3	3
2	Facilidad de acceso	Si= 1/No= 0	1	1	1

Matriz de evaluación de calidad para la característica: Usabilidad.

	Características /Subcaracterísticas	Métrica	OpenStack	OpenNebula	Eucalyptus
	USABILIDAD				
	CAPACIDAD PARA SER APRENDIDO				
	Fácil de aprender				
1	Material de aprendizaje de acceso libre	Si= 1/No= 0	1	1	1
2	Cursos on-line	Si= 1/No= 0	1	0	0
	CAPACIDAD PARA SER OPERADO				
	Capacidad de operación				
1	Apariencia de la consola de administración	[0:4]	4	1	1
2	Fácil administración	[0:4]	3	2	2
3	Acceso por red privada	Si= 1/No= 0	1	1	1
	ANÁLISIS DE DOCUMENTACION				
	Análisis de documentación				
1	Manuales de instalación y de usuario	Si= 1/No= 0	1	1	1

Matriz de evaluación de calidad para la característica: Seguridad.

	Características /Subcaracterísticas	Métrica	OpenStack	OpenNebula	Eucalyptus
	SEGURIDAD				
	VULNERABILIDAD				
	Vulnerabilidad				
1	Acceso autorizado	Si= 1/No= 0	1	1	0
2	Control sobre el contenido de cada usuario	Si= 1/No= 0	1	0	0
3	Gestionada por terceros	Si= 1/No= 0	1	0	1

Matriz de evaluación de calidad de la característica: Interoperabilidad.

	Características /Subcaracterísticas	Métrica	OpenStack	OpenNebula	Eucalyptus
	INTEROPERABILIDAD				
	INTEROPERABILIDAD				
	Interoperabilidad con Servicios Web				
1	Integración con componentes AWS	Si= 1/No= 0	1	1	1
2	Integración con componentes Windows Azure.	Si= 1/No= 0	1	1	1

ANEXO C: MANUAL DE OPERACIÓN DE OPENSTACK

Crear Proyectos en OpenStack

```
openstack project create --domain default --description "Test SDN" sdn
```

```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Test SDN                         |
| domain_id   | default                          |
| enabled     | True                             |
| id          | 3b6e9ac2fa844bb59ec0f1e0ab9dc5bc |
| is_domain   | False                            |
| name        | sdn                              |
| parent_id   | default                          |
+-------------+----------------------------------+
```

```
openstack project list
```

```
+----------------------------------+----------+
| ID                               | Name     |
+----------------------------------+----------+
| 08d5acf0e37d48deb2a06a87386d3142 | demo     |
| 1343580ad1674010ae262b34657db670 | admin    |
| 3b6e9ac2fa844bb59ec0f1e0ab9dc5bc | sdn      |
| b19ec180300443c2a711beffe6a982e5 | services |
+----------------------------------+----------+
```

Crear usuarios OpenStack

```
openstack user create --domain default --project sdn --password
sdnpass sdnadmin
```

```
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| default_project_id  | 3b6e9ac2fa844bb59ec0f1e0ab9dc5bc |
| domain_id           | default                          |
| enabled             | True                             |
| id                  | 35553f404d17436cac6ad30a007381c9 |
| name                | sdnadmin                         |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

```
openstack role add –project sdn –user sdnadmin admin
```

```
[root@controller0 ~(keystone_admin)]# openstack role add --project sdn --user sdn
admin admin
[root@controller0 ~(keystone_admin)]#
```

Descubir Nodos de Cómputo desde el Nodo de Control

```
su -s /bin/bash nova -c "nova-manage cell_v2 discover_hosts"
openstack compute service list
```

```
+----+------------------+-------------+----------+---------+-------+----------------------------+
| ID | Binary           | Host        | Zone     | Status  | State | Updated At                 |
+----+------------------+-------------+----------+---------+-------+----------------------------+
|  3 | nova-cert        | controller0 | internal | enabled | up    | 2018-01-22T12:57:46.000000 |
|  4 | nova-conductor   | controller0 | internal | enabled | up    | 2018-01-22T12:57:41.000000 |
|  5 | nova-scheduler   | controller0 | internal | enabled | up    | 2018-01-22T12:57:39.000000 |
|  6 | nova-consoleauth | controller0 | internal | enabled | up    | 2018-01-22T12:57:45.000000 |
|  7 | nova-compute     | compute2    | nova     | enabled | up    | 2018-01-22T12:57:41.000000 |
|  8 | nova-compute     | compute1    | nova     | enabled | up    | 2018-01-22T12:57:41.000000 |
+----+------------------+-------------+----------+---------+-------+----------------------------+
```

Crear imágenes para máquinas virtuales

```
wget http://download.cirros-cloud.net/0.4.0/cirros-0.4.0-x86_64-disk.img -P /var/kvm/images
```

```
[root@controller0 ~(keystone_admin)]# wget http://download.cirros-cloud.net/0.4.0/cirros-0.4.0-x86_64-disk.img
-P /var/kvm/images
--2018-01-22 08:11:40--  http://download.cirros-cloud.net/0.4.0/cirros-0.4.0-x86_64-disk.img
Resolviendo download.cirros-cloud.net (download.cirros-cloud.net)... 64.90.42.85, 2607:f298:6:a036::bd6:a72a
Conectando con download.cirros-cloud.net (download.cirros-cloud.net)[64.90.42.85]:80... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 12716032 (12M) [text/plain]
Grabando a: “/var/kvm/images/cirros-0.4.0-x86_64-disk.img”

100%[=====================================================================>] 12.716.032   534KB/s   en 25s

2018-01-22 08:12:05 (503 KB/s) - “/var/kvm/images/cirros-0.4.0-x86_64-disk.img” guardado [12716032/12716032]
```

```
openstack image create "cirros-0.4.0" --file /var/kvm/images/cirros-0.4.0-x86_64-disk.img --disk-format qcow2 --container-format bare –public
```

```
+------------------+------------------------------------------------------+
| Field            | Value                                                |
+------------------+------------------------------------------------------+
| checksum         | 443b7623e27ecf03dc9e01ee93f67afe                     |
| container_format | bare                                                 |
| created_at       | 2018-01-22T13:16:35Z                                 |
| disk_format      | qcow2                                                |
| file             | /v2/images/a635e4f0-ae50-4097-bfef-55a17938f568/file |
| id               | a635e4f0-ae50-4097-bfef-55a17938f568                 |
| min_disk         | 0                                                    |
| min_ram          | 0                                                    |
| name             | cirros-0.4.0                                         |
| owner            | 1343580ad1674010ae262b34657db670                     |
| protected        | False                                                |
| schema           | /v2/schemas/image                                    |
| size             | 12716032                                             |
| status           | active                                               |
| tags             |                                                      |
| updated_at       | 2018-01-22T13:16:36Z                                 |
| virtual_size     | None                                                 |
| visibility       | public                                               |
+------------------+------------------------------------------------------+
```

```
openstack image list
```

```
+--------------------------------------+--------------+--------+
| ID                                   | Name         | Status |
+--------------------------------------+--------------+--------+
| 4cb46a2c-086f-4ef3-b007-bbf55cb811af | cirros       | active |
| a635e4f0-ae50-4097-bfef-55a17938f568 | cirros-0.4.0 | active |
+--------------------------------------+--------------+--------+
```

Mostrar estado de los Agentes de Red de Neutron

```
openstack network agent list
```

```
+--------------------------------------+---------------+------------+-------------------+-------+-------+----------------------------+
| ID                                   | Agent Type    | Host       | Availability Zone | Alive | State | Binary                     |
+--------------------------------------+---------------+------------+-------------------+-------+-------+----------------------------+
| 4c772f09-a155-4763-9a74-58471bc85902 | Metadata agent | controller0 | None              | True  | UP    | neutron-metadata-agent     |
| 718cf188-a00c-49e8-aa3e-54c36141a8bc | DHCP agent    | controller0 | nova              | True  | UP    | neutron-dhcp-agent         |
| a45755be-b396-4aa1-904c-4d5d8e6e157e | ODL L2        | controller0 | None              | True  | UP    | neutron-odlagent-portbinding |
| b65995e9-9ac2-4ee8-a236-4667779c4fdb | ODL L2        | compute2    | None              | True  | UP    | neutron-odlagent-portbinding |
| f2136b1a-f7cc-49de-9433-e6ffc939203d | ODL L2        | compute1    | None              | True  | UP    | neutron-odlagent-portbinding |
+--------------------------------------+---------------+------------+-------------------+-------+-------+----------------------------+
```

```
openstack network list
```

```
+--------------------------------------+---------+--------------------------------------+
| ID                                   | Name    | Subnets                              |
+--------------------------------------+---------+--------------------------------------+
| 5a1542dc-4150-4cd1-9226-43daea5f6e89 | privada | 5a12840a-a1c4-4456-ac95-1c03e20f9713 |
| bd78b613-487a-4c4e-b6bd-0ec8b8284846 | ext_net | 21302cee-ba4c-410e-bcf7-ec0e6c6b6d4b |
+--------------------------------------+---------+--------------------------------------+
```

```
openstack subnet list
```

```
+--------------------------------------+---------+--------------------------------------+--------------+
| ID                                   | Name    | Network                              | Subnet       |
+--------------------------------------+---------+--------------------------------------+--------------+
| 21302cee-ba4c-410e-bcf7-ec0e6c6b6d4b | subnet2 | bd78b613-487a-4c4e-b6bd-0ec8b8284846 | 10.24.8.0/24 |
| 5a12840a-a1c4-4456-ac95-1c03e20f9713 | subnet1 | 5a1542dc-4150-4cd1-9226-43daea5f6e89 | 10.0.0.0/24  |
+--------------------------------------+---------+--------------------------------------+--------------+
```

Creación de redes con Neutron

```
openstack router create router-sdn
```

```
+-------------------------+--------------------------------------+
| Field                   | Value                                |
+-------------------------+--------------------------------------+
| admin_state_up          | UP                                   |
| availability_zone_hints | None                                 |
| availability_zones      | None                                 |
| created_at              | 2018-01-22T13:29:13Z                 |
| description             |                                      |
| distributed             | False                                |
| external_gateway_info   | None                                 |
| flavor_id               | None                                 |
| ha                      | False                                |
| id                      | 30cf95d9-9c72-4e13-b696-09928ee8273b |
| name                    | router-sdn                           |
| project_id              | 1343580ad1674010ae262b34657db670     |
| revision_number         | None                                 |
| routes                  |                                      |
| status                  | ACTIVE                               |
| updated_at              | 2018-01-22T13:29:13Z                 |
+-------------------------+--------------------------------------+
```

```
openstack network create int_sdn --provider-network-type vxlan
```

Field	Value
admin_state_up	UP
availability_zone_hints	
availability_zones	
created_at	2018-01-22T13:30:18Z
description	
dns_domain	None
id	0f7bf36f-c0a2-49d8-9afe-c1056ab86ffc
ipv4_address_scope	None
ipv6_address_scope	None
is_default	None
mtu	1450
name	int_sdn
port_security_enabled	True
project_id	1343580ad1674010ae262b34657db670
provider:network_type	vxlan
provider:physical_network	None
provider:segmentation_id	79
qos_policy_id	None
revision_number	3
router:external	Internal
segments	None
shared	False
status	ACTIVE
subnets	
updated_at	2018-01-22T13:30:18Z

```
openstack subnet create subnet-sdn1 --network int_sdn \
--subnet-range 50.0.0.0/24 --gateway 50.0.0.1 \
--dns-nameserver 8.8.8.8
```

Field	Value
allocation_pools	50.0.0.2-50.0.0.254
cidr	50.0.0.0/24
created_at	2018-01-22T13:31:47Z
description	
dns_nameservers	8.8.8.8
enable_dhcp	True
gateway_ip	50.0.0.1
host_routes	
id	2f3f99aa-1025-4896-b1b0-89c75ffeb02f
ip_version	4
ipv6_address_mode	None
ipv6_ra_mode	None
name	subnet-sdn1
network_id	0f7bf36f-c0a2-49d8-9afe-c1056ab86ffc
project_id	1343580ad1674010ae262b34657db670
revision_number	2
segment_id	None
service_types	
subnetpool_id	None
updated_at	2018-01-22T13:31:47Z

```
openstack router add subnet router-sdn subnet-sdn1
openstack network create \
```

```
--provider-physical-network extnet \
--provider-network-type flat --external ext_net
```

```
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| availability_zone_hints   |                                      |
| availability_zones        |                                      |
| created_at                | 2018-01-08T21:53:17Z                 |
| description               |                                      |
| id                        | 2f438218-17e3-4af7-bfc0-da9ddb46f213 |
| ipv4_address_scope        |                                      |
| ipv6_address_scope        |                                      |
| is_default                | False                                |
| mtu                       | 1450                                 |
| name                      | ext_net                              |
| project_id                | f9326e7231e84ad1902787e500ab7410     |
| provider:network_type     | vxlan                                |
| provider:physical_network |                                      |
| provider:segmentation_id  | 29                                   |
| revision_number           | 3                                    |
| router:external           | True                                 |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tags                      |                                      |
| tenant_id                 | f9326e7231e84ad1902787e500ab7410     |
| updated_at                | 2018-01-08T21:53:17Z                 |
+---------------------------+--------------------------------------+
```

```
openstack subnet create subnet-sdn2 \
--network ext_net --subnet-range 10.24.8.0/24 \
--allocation-pool start=10.24.8.200,end=10.24.8.254 \
--gateway 10.24.8.2 --dns-nameserver 8.8.8.8 --no-dhcp
```

```
+-------------------+-----------------------------------------------+
| Field             | Value                                         |
+-------------------+-----------------------------------------------+
| allocation_pools  | {"start": "10.24.8.135", "end": "10.24.8.150"} |
| cidr              | 10.24.8.0/24                                  |
| created_at        | 2018-01-08T22:04:23Z                          |
| description       |                                               |
| dns_nameservers   |                                               |
| enable_dhcp       | False                                         |
| gateway_ip        | 10.24.8.2                                     |
| host_routes       |                                               |
| id                | 6a60c66a-46af-482b-bd3d-f2bcb76b0057          |
| ip_version        | 4                                             |
| ipv6_address_mode |                                               |
| ipv6_ra_mode      |                                               |
| name              |                                               |
```

```
openstack router set router-sdn --external-gateway ext_net
```

Creación de Instancias

```
Int_Net_ID=`openstack network list | grep int_net | awk '{print $2}'`
openstack server create --flavor m1.nano --image cirros-0.4.0 --
security-group default --nic net-id=$Int_Net_ID sdn-vm
```

Field	Value
OS-DCF:diskConfig	MANUAL
OS-EXT-AZ:availability_zone	
OS-EXT-SRV-ATTR:host	None
OS-EXT-SRV-ATTR:hypervisor_hostname	None
OS-EXT-SRV-ATTR:instance_name	
OS-EXT-STS:power_state	NOSTATE
OS-EXT-STS:task_state	scheduling
OS-EXT-STS:vm_state	building
OS-SRV-USG:launched_at	None
OS-SRV-USG:terminated_at	None
accessIPv4	
accessIPv6	
addresses	
adminPass	F9tqv4W2Wm5h
config_drive	
created	2018-01-22T13:43:29Z
flavor	m1.nano (6)
hostId	
id	85b4eb68-1a29-4987-9630-80634d37f30a
image	cirros-0.4.0 (a635e4f0-ae50-4097-bfef-55a17938f568)
key_name	None
name	sdn-vm
progress	0
project_id	1343580ad1674010ae262b34657db670
properties	
security_groups	name='default'
status	BUILD
updated	2018-01-22T13:43:29Z
user_id	6e75e391658245c8b9624a1235a11264
volumes_attached	

```
openstack server list
```

ID	Name	Status	Networks	Image Name
85b4eb68-1a29-4987-9630-80634d37f30a	sdn-vm	ACTIVE	int_sdn=50.0.0.5	cirros-0.4.0
ec1a2339-551d-4ab5-b6af-85d8a6855ce5	vm2	ACTIVE	privada=10.0.0.12, 10.24.8.137	cirros
d1dd20f7-7248-4857-a2d7-8c4e6b3b59c1	vm1	ACTIVE	privada=10.0.0.11, 10.24.8.140	cirros

```
openstack floating ip create ext_net
```

Field	Value
created_at	2018-01-22T13:48:50Z
description	
fixed_ip_address	None
floating_ip_address	10.24.8.147
floating_network_id	bd78b613-487a-4c4e-b6bd-0ec8b8284846
id	93df61a6-4489-48e4-965c-924c47a2205c
name	None
port_id	None
project_id	1343580ad1674010ae262b34657db670
revision_number	1
router_id	None
status	ACTIVE
updated_at	2018-01-22T13:48:50Z

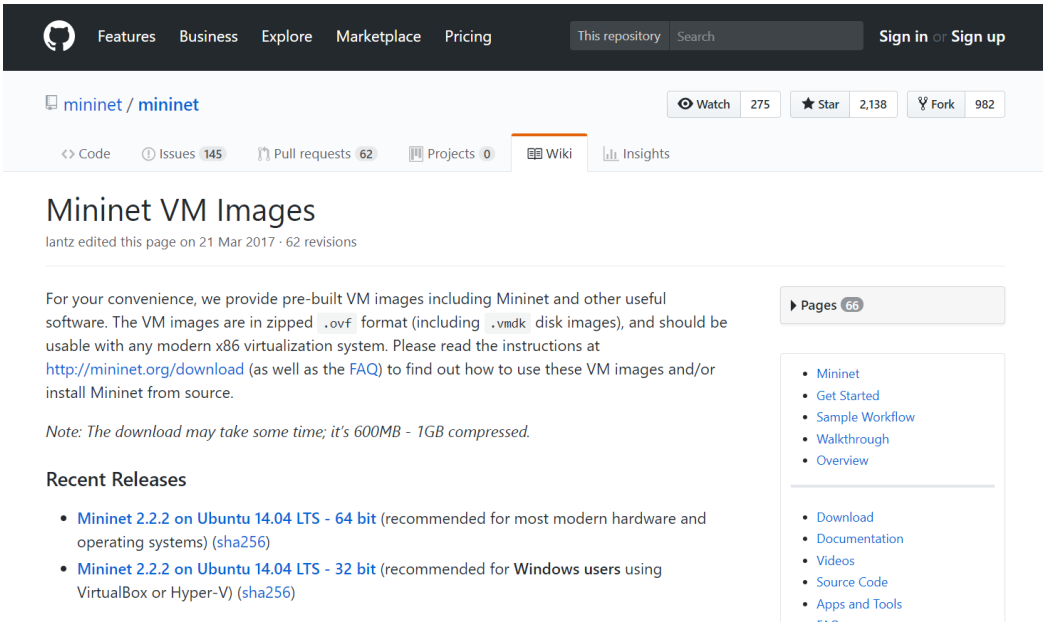
```
openstack server add floating ip sdn-vm 10.24.8.147
```

ANEXO D: INSTALACIÓN DE CONTROLADORES OPENFLOW

Instalación de Mininet

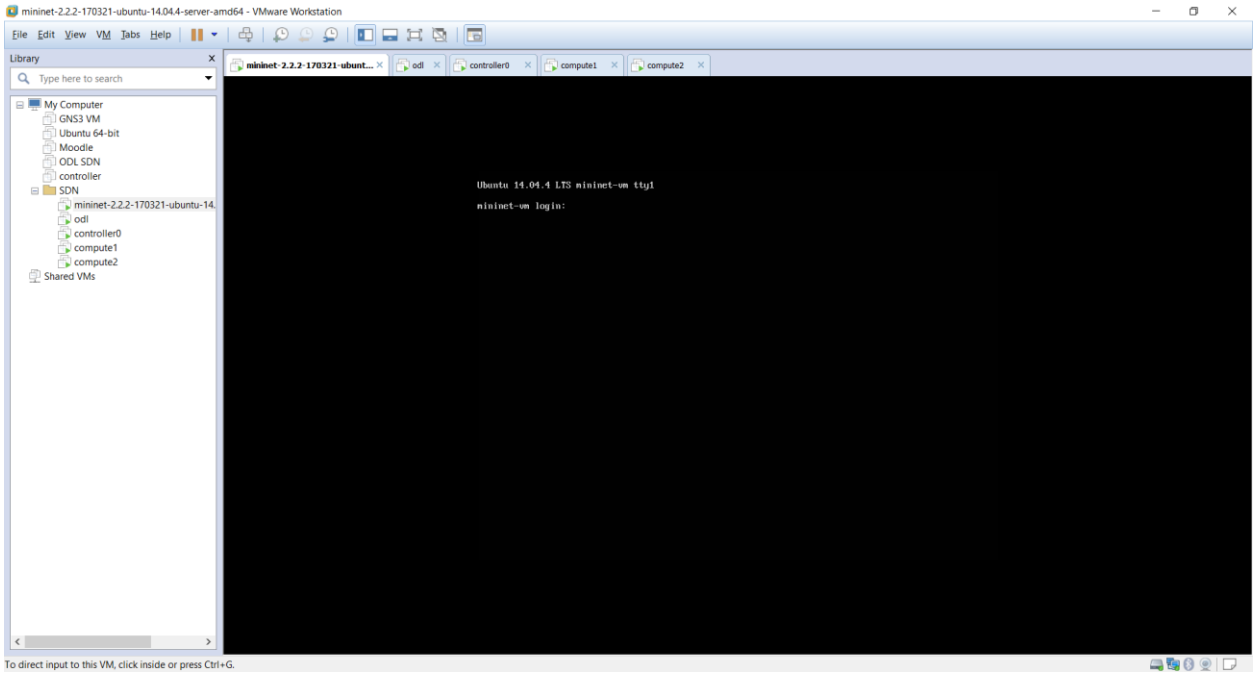
La forma más fácil de instalar mininet es usando el código proporcionado por los desarrolladores:

Descargamos la imagen de mininet de https://github.com/mininet/mininet/wiki/Mininet-VM-Images



Instalar cualquier sistema de virtualización como VMware o VirtualBox, aunque el segundo es más lento que VMware en el desarrollo de pruebas.

Cargar la máquina virtual en el virtualizador y arrancar el software. El usuario y la contraseña son mininet/mininet.

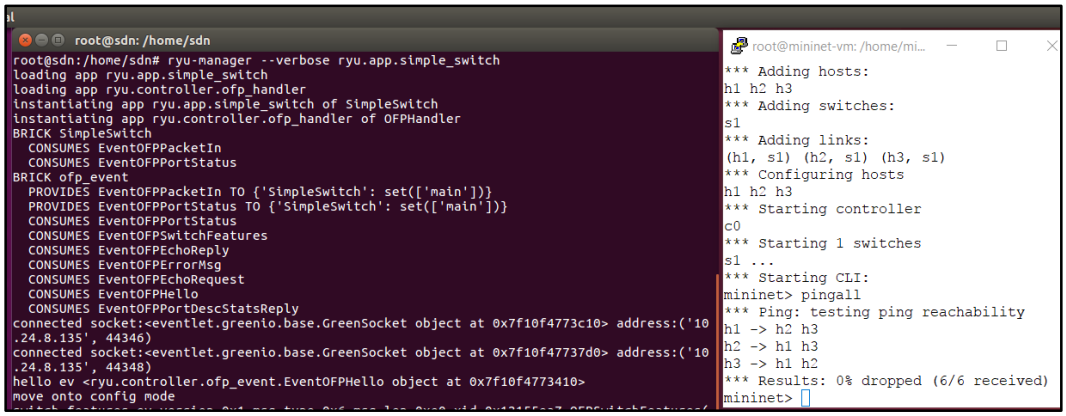


Instalación Controlador RYU

Para instalar RYU es necesario contar con la herramienta Python-pip. Luego se ejecutan las siguientes instrucciones sobre cualquier distribución de linux.

```
pip install ryu
git clone git://github.com/osrg/ryu.git
cd ryu; python ./setup.py install
```

Una vez que RYU se encuentra correctamente compilado generara un entorno de línea de comandos sobre la cual se pueden ejecutar varias combinaciones de instrucciones.



Instalación Controlador Floodlight

Floodlight funciona en cualquier distribución de linux que tenga instalado Java Development Kit en su versión más estable para este caso JDK 8 sobre linux ubuntu. Se recomienda usar la versión Floodlight Master o una superior.

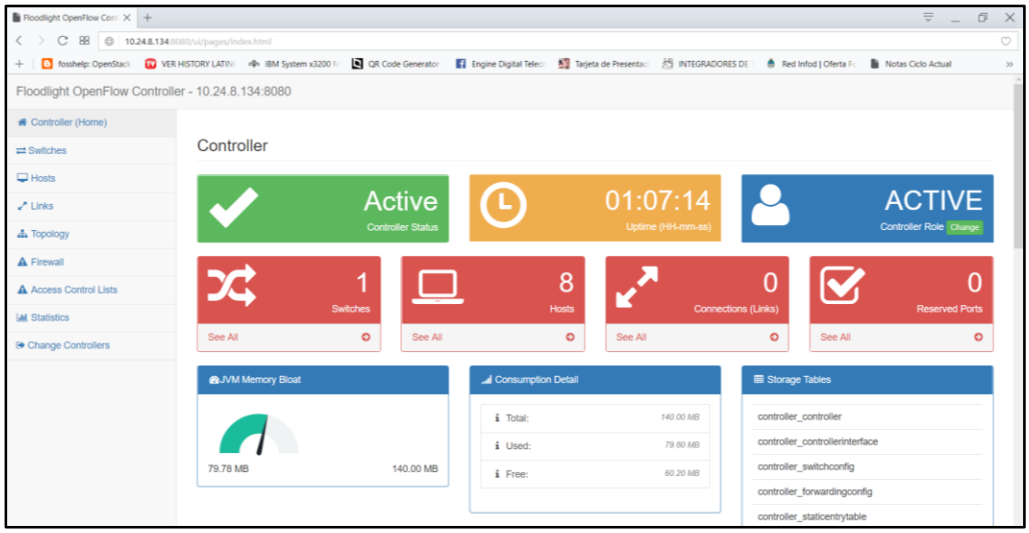
```
sudo apt-get install build-essential ant maven python-dev
```

Floodlight es simple de descargar y compilar desde Github. Se sigue los siguientes pasos para descargar e instalar una nueva copia de Floodlight o actualizar una instalación existente de Floodlight:

```
git clone git://github.com/floodlight/floodlight.git
cd floodlight
git submodule init
git submodule update
ant
sudo mkdir /var/lib/floodlight
sudo chmod 777 /var/lib/floodlight
```

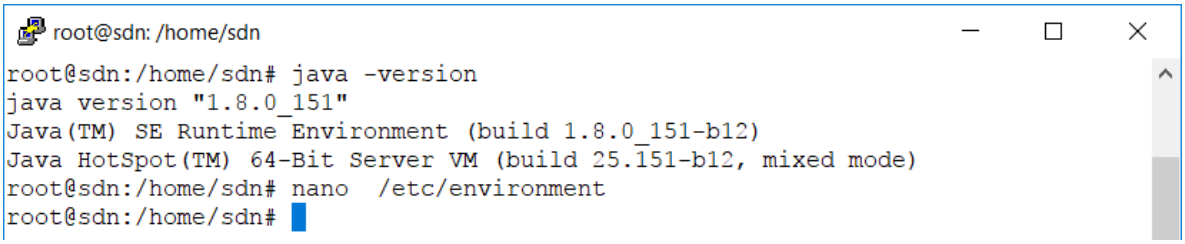
Suponiendo que java está correctamente instalado, se puede ejecutar directamente el archivo floodlight.jar producido por ant desde dentro del directorio de Floodlight:

```
java -jar target/floodlight.jar
```



Instalación Controlador OpenDaylight

Para instalar el controlador OpenDaylight lo primero es descargar el paquete de la página oficial del Proyecto http://www.opendaylight.org/software/downloads luego se debe obtener la versión más estable de Java para generar la máquina virtual de Karaf.

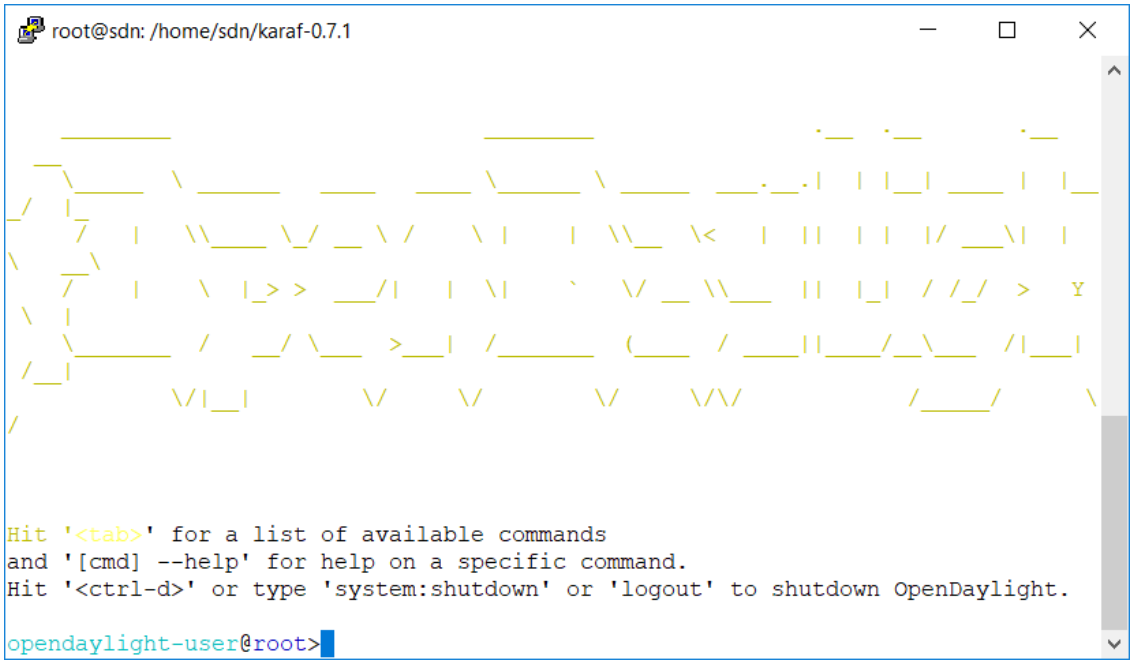


La distribución Karaf no tiene características habilitadas por defecto. Sin embargo, todas las características están disponibles para ser instaladas. Por razones de compatibilidad, no puede

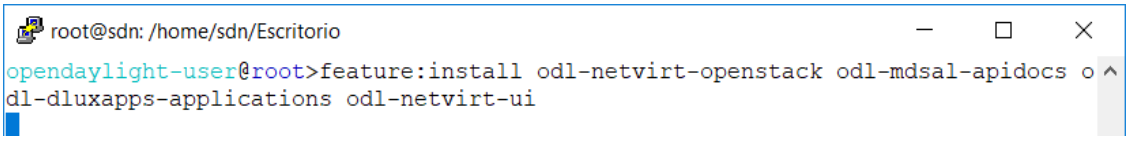
habilitar todas las funciones simultáneamente. Intentamos documentar las incompatibilidades conocidas en la sección Instalar características de Karaf a continuación.

Para ejecutar la distribución de Karaf:

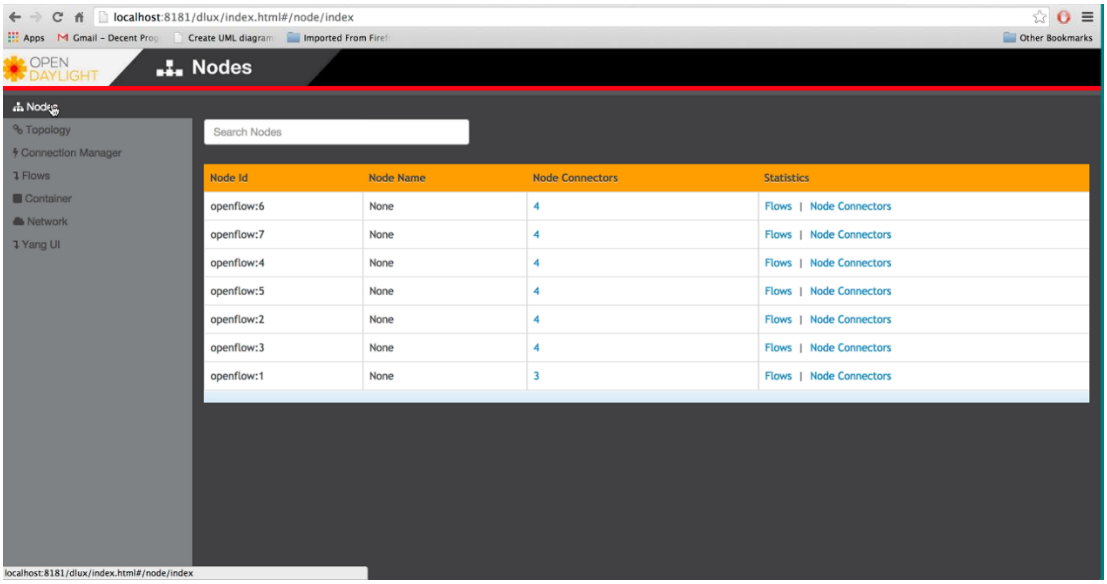
- Descomprime el archivo zip.
- Navega al directorio.
- ejecutar ./bin/karaf.



Para instalar características se sigue el formato feature: caracteristica1 caracteristica2



Para acceder a la interfaz gráfica se usa la url http://IP-CONTROLADOR/dlux/index.html



Instalación Controlador ONOS

La siguiente sección describe cómo instalar ONOS en una única máquina de destino, ejecutando los pasos de instalación localmente (en la propia máquina de destino).

Los paquetes predeterminados de ONOS suponen que ONOS se instala bajo / opt , así que primero asegurarse de que el directorio exista y acceder a él.

```
sudo mkdir /opt
cd /opt
```

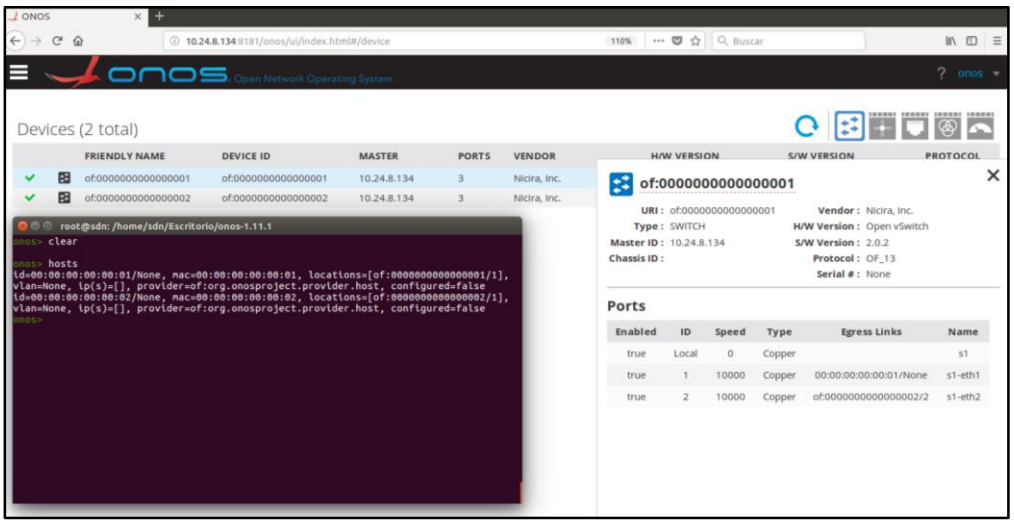
Descargar la versión deseada de ONOS en formato tar.gz. Se dirige al sitio web de ONOS, sección de descargas (http://downloads.onosproject.org). Se descarga el archivo y se ubica en /opt.

```
sudo wget -c http://downloads.onosproject.org/release/onos-$ONOS_VERSION.tar.gz
```

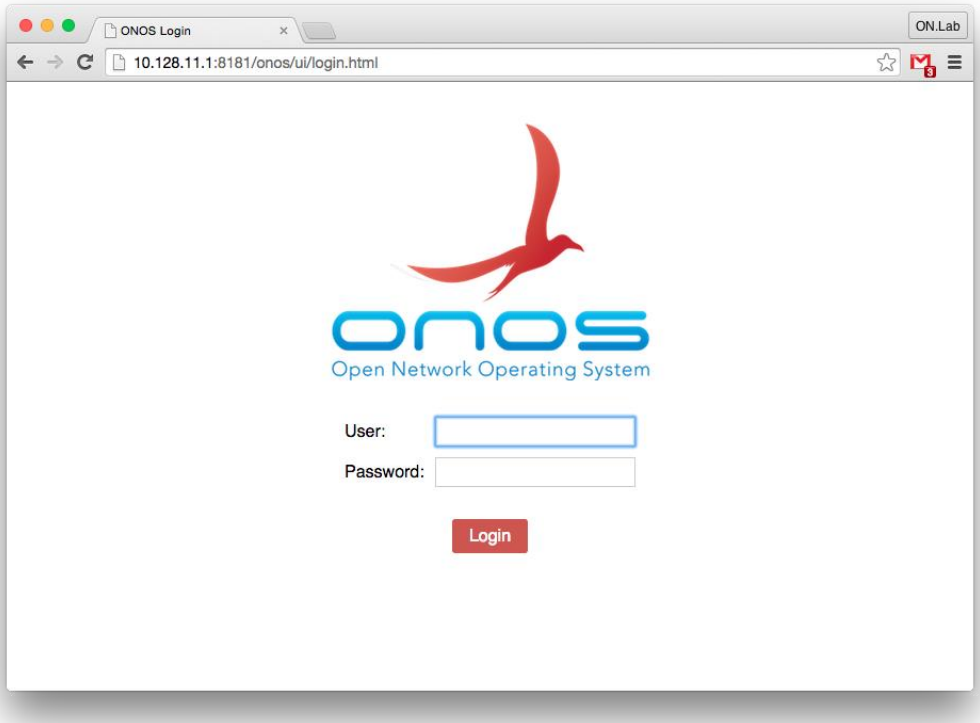
Se descomprime el archivo y se arranca el script ejecutable.

```
tar xzf onos-$ONOS_VERSION.tar.gz

sudo mv onos-$ONOS_VERSION onos
```



Para acceder a la interfaz gráfica se usa la url http://IP-CONTROLADOR/onos/ui/login.html



ANEXO E: TABLAS DE FLUJO DEL MECANISMO NETVIRT