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.

Carlos E.Sandoval
Facultad de Ingeniería en Ciencias Aplicadas
Universidad Técnica del Norte
Ibarra, Ecuador
cesandovalc@utn.edu.ec

Carlos A. Vásquez
Facultad de Ingeniería en Ciencias Aplicadas
Universidad Técnica del Norte
Ibarra, Ecuador
cavasquez@utn.edu.ec

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.

Palabras Clave—sdn; openflow; nfv; openvswitch; controller; cluster; agent; network; neutron; cloud; opensource; linux

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.

Keywords—sdn; openflow; nfv; openvswitch; controller; cluster; agent; network; neutron; cloud; opensource; linux

I. INTRODUCCION

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. [1]

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.

II. REDES DEFINIDAS POR SOFTWARE

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. [2]

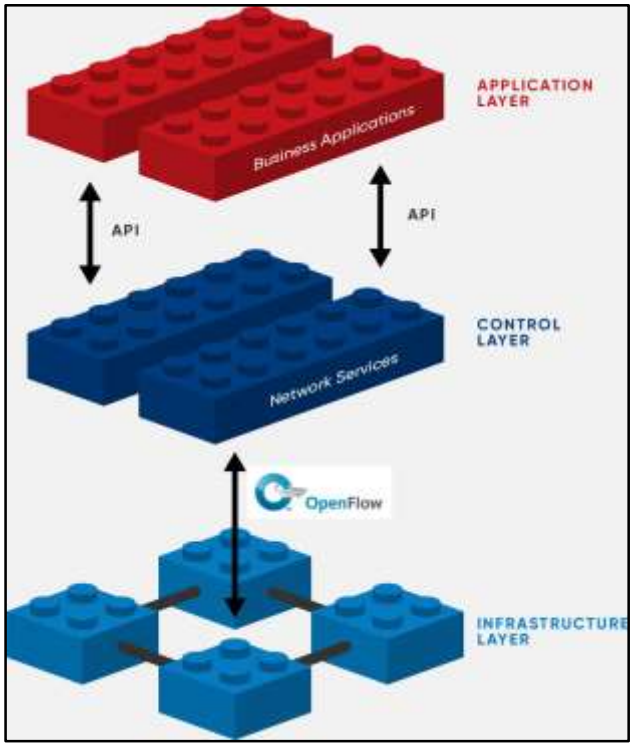

Fig. 1. Arquitectura y elementos de SDN

III. 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.

A. PLANO DE CONTROL

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).

B. 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.

C. 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. [3]

IV. 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. [4]

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. [5]

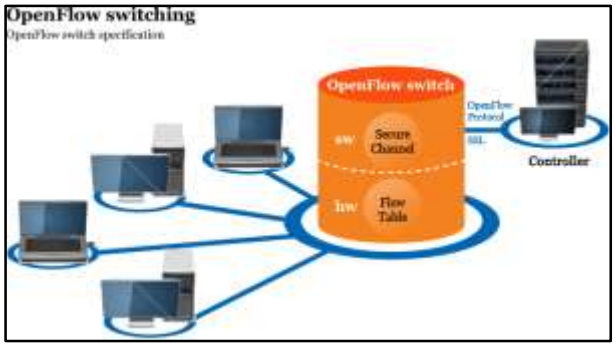

Fig. 2. Openflow Switching.

V. CONTROLADORES PARA SDN

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. [4] [7]

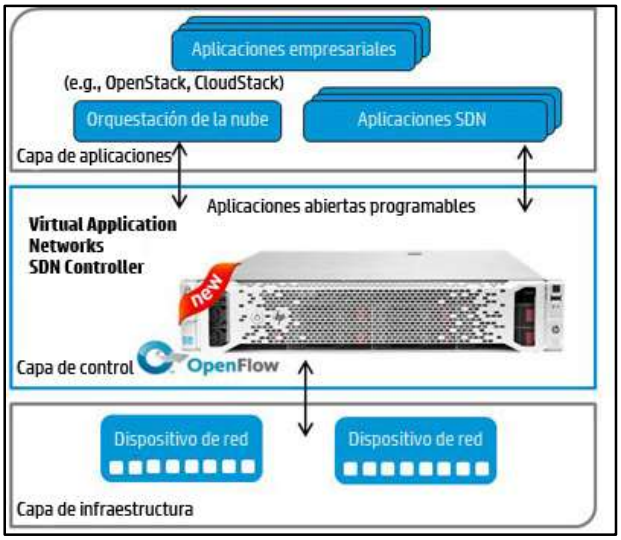

Fig. 3. Controlador SDN HP

VI. 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. [8]

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 . A esto se le conoce como "Network Cloud". [5]

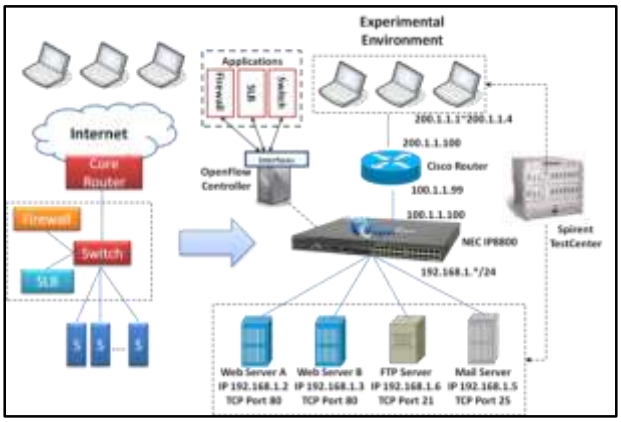

Fig. 4. Arquitectura simplificada de una red de computación en la nube.

VII. CLUSTERING

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. [6]

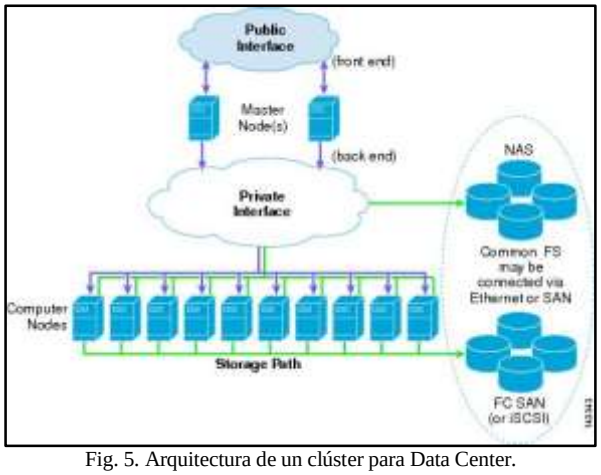

Fig. 5. Arquitectura de un clúster para Data Center.

VIII. SITUACION ACTUAL DATACENTER FICA

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.

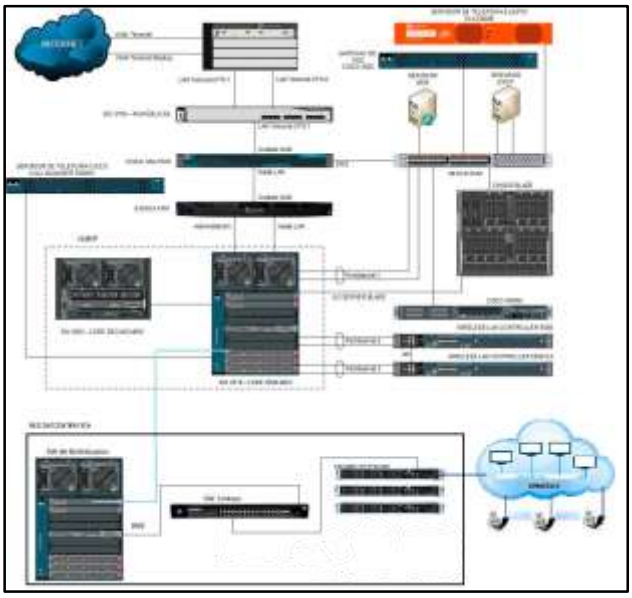

Fig. 6. Infraestructura física de la red UTN

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 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. [7]

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.

Después de construir el modelo de evaluación y generar las matrices 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. 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.

TABLA I
Resultados del análisis de calidad del software según la norma ISO/IEC/IEEE 2504n

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

IX. ACONDICIONAMIENTO DE LA PLATAFORMA

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. [8]

Esta 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.

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.

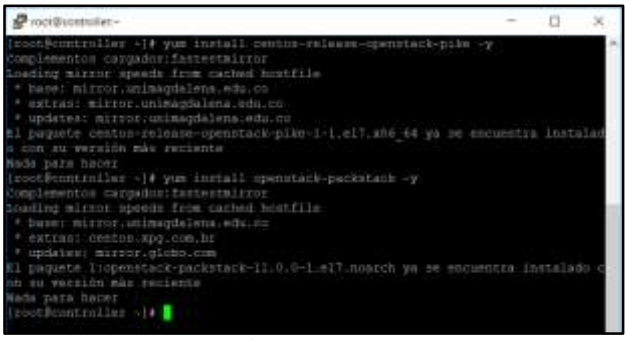

Fig. 7. Instalación del repositorio de OpenStack

Se modifican los campos necesarios y más importantes para proceder 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.

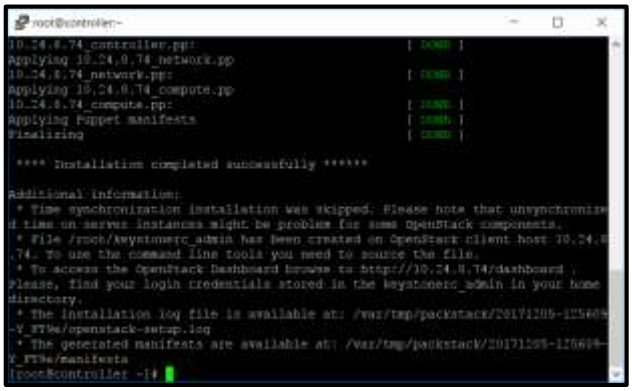

Fig. 8. Instalación exitosa de OpenStack.

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.

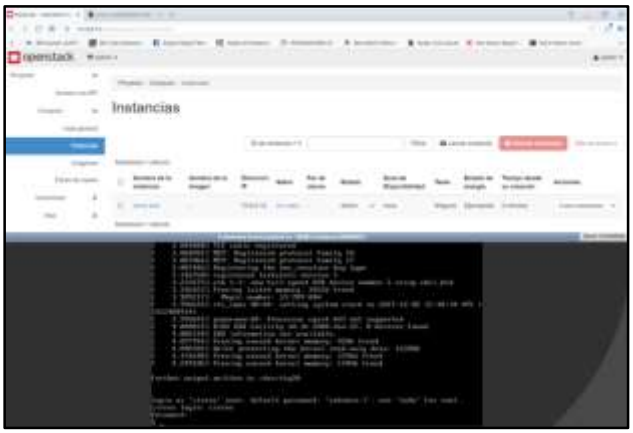

Fig. 9. Instancia funcionando sobre OpenStack.

En este apartado se configura también 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): 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.

```
+-------+-------+---------------------------+
| Alive | State | Binary                    |
+-------+-------+---------------------------+
| :-)   | UP    | neutron-vpn-agent         |
| :-)   | UP    | neutron-dhcp-agent        |
| :-)   | UP    | neutron-metering-agent    |
| :-)   | UP    | neutron-metadata-agent    |
| :-)   | UP    | neutron-openvswitch-agent |
| :-)   | UP    | neutron-lbaasv2-agent     |
+-------+-------+---------------------------+
```

Fig. 10. Agentes del módulo de red Neutron.

X. EVALUACION 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.

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.

No todos los controladores existentes 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

A. PRUEBA CON RYU

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.

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. [12]

Fig. 11. Comunicación establecida entre mininet y RYU.

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.

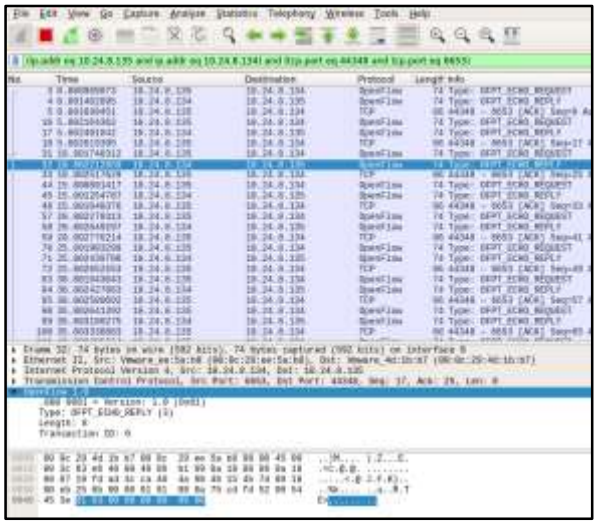

Fig. 12. Captura de tráfico OpenFlow entre RYU y mininet.

B. 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.

Fig. 13. Interfaz de usuario de Floodlight.

Una prueba básica se puede realizar usando el simulador de redes mininet. Este simulador trabaja a base de scripts de ejecución o comandos parametrizados.

Fig. 14. Creación de una red OpenFlow de prueba.

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.

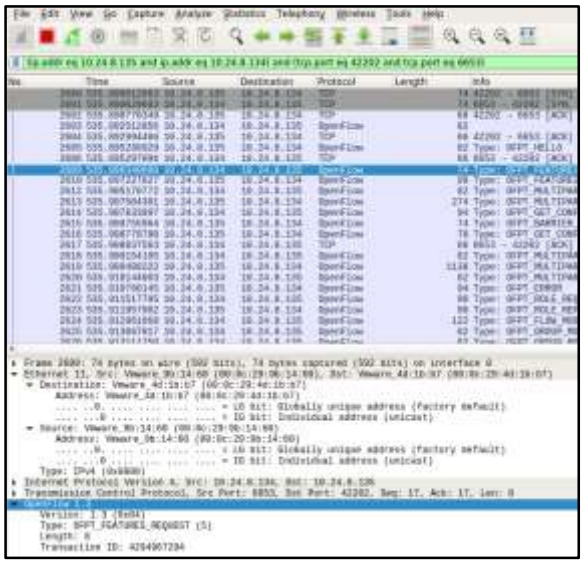
Fig. 15. Captura de tráfico sobre la interfaz del controlador Floodlight.

C. PRUEBA CON OPENDAYLIGHT

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.

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. [10]

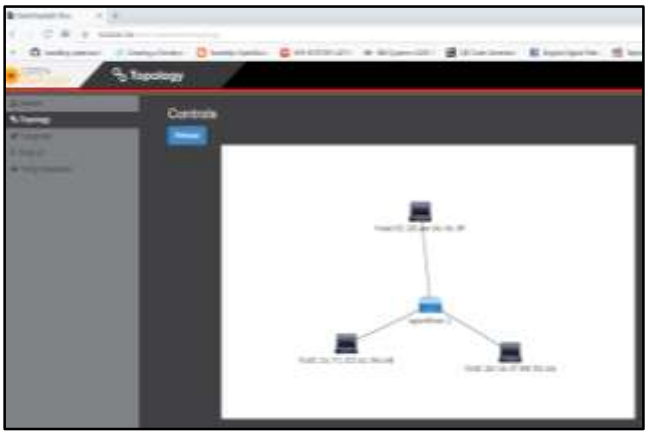

Fig. 16. Interfaz DLUX de OpenDaylight.

Una vez que el controlador y sus módulos se encuentran activos, se procede a generar tráfico 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 se filtra los mensajes para observar únicamente los pertenecientes a la conversación OpenFlow.

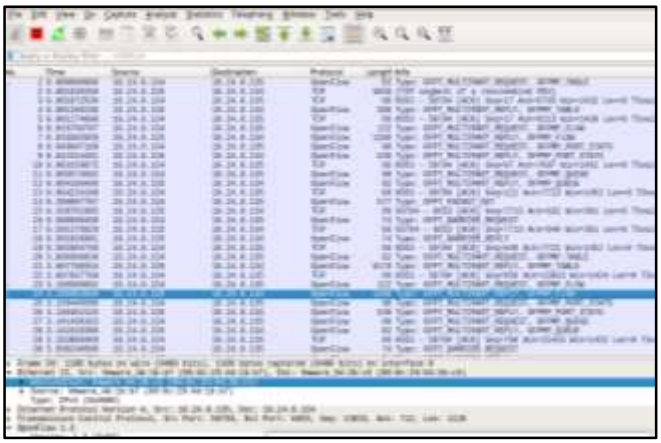
Fig. 17. Sesión OpenFlow de OpenDaylight en Wireshark

D. 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.

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. [13]

Fig. 18. Descubrimiento de hosts en ONOS Controller

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

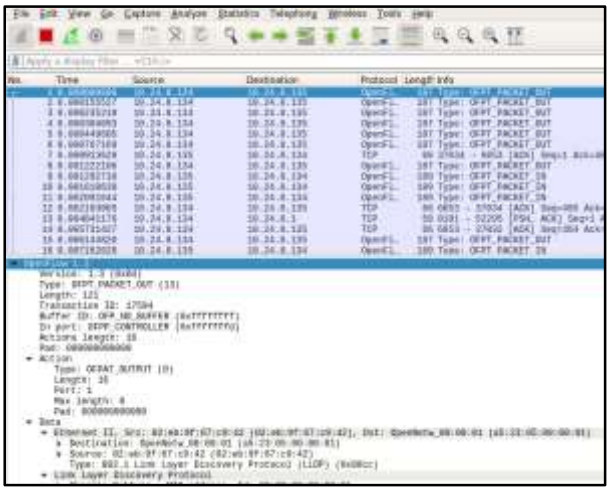

Fig. 19. Mensaje PACKET_OUT generado por ONOS hacia el conmutador simulado.

XI. 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)".

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.

- Propósito
- Alcance
- Perspectiva del producto
- Características de los usuarios
- Requisitos funcionales
- Requisitos no funcionales
- Requisitos de rendimiento

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.

XII. IMPLEMENTACION DEL CONTROLADOR SDN

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.

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 correctamente

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.

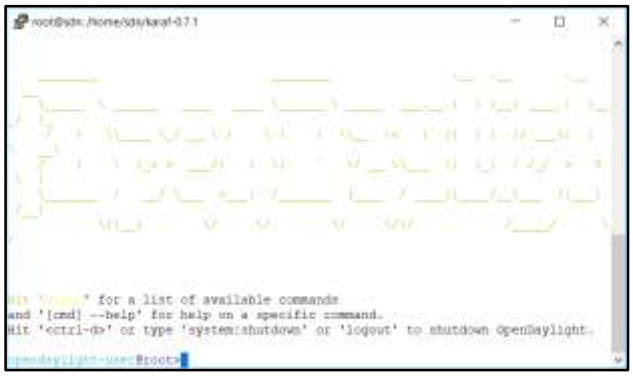

Fig. 20. Consola de configuración de OpenDaylight.

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.

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 para la 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.

Fig. 21. Entorno grafico de los módulos NetVirt y DLUX

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.

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. Esto confirma que el agente ODL se encuentra integrado con OpenStack.

```
nstack network agent list
+--------------------+------------+--------------------+-------+-------+
| Agent Type         | Host       | Availability Zone  | Alive | State |
+--------------------+------------+--------------------+-------+-------+
| ODL L2             | controller | None               | True  | UP    |
| L3 agent           | controller | nova               | True  | UP    |
| DHCP agent         | controller | nova               | True  | UP    |
| Metadata agent     | controller | None               | True  | UP    |
+--------------------+------------+--------------------+-------+-------+
```

Fig. 22. Agente de red ODL L2

XIII. PRUEBAS DE FUNCIONAMIENTO

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. 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.

Fig.23. Tablas de flujo reflejadas en ODL.

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.

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.

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.

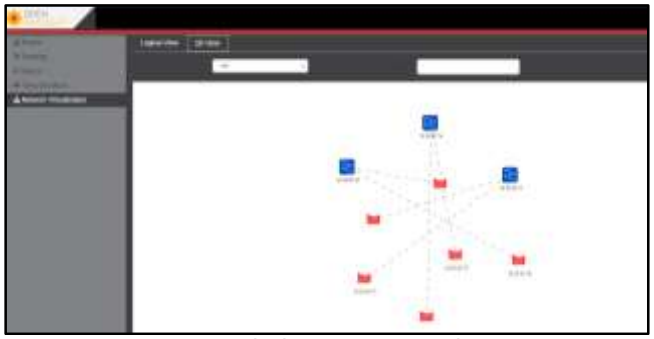

Fig. 24. Topología lógica de la red del clúster SDN.

La comunicación entre los nodos usa el método de tunelizacion para redes overlay, 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.

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.

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.

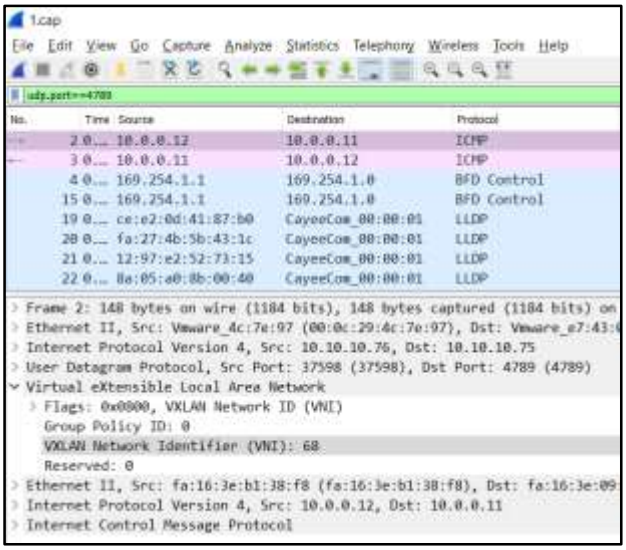

Fig. 25. Tráfico VXLAN decodificado en Wireshark.

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.

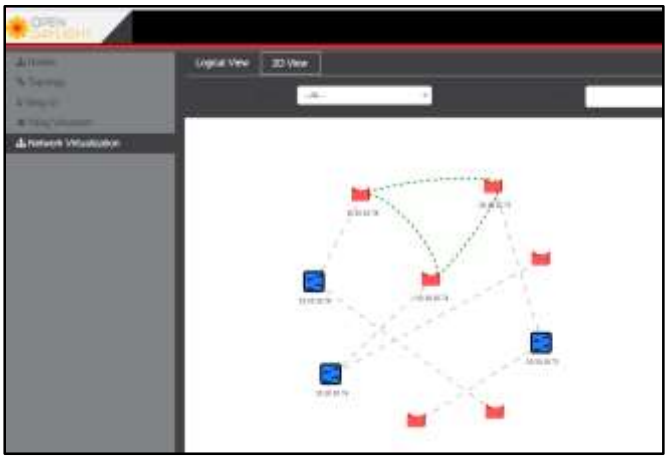

Fig. 26. Anillo de túneles VXLAN creados por ITM.

XIV. CONCLUSIONES

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.

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 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.

XV. AGRADECIMIENTO

A la Universidad Técnica del Norte y a la Facultad de Ingeniería en Ciencias Aplicadas, a toda la planta docente que durante tanto tiempo me ha preparado para enfrentar los retos de la vida profesional.

Un sincero y especial agradecimiento a mi director, 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 proyecto imposible y que, con trabajo duro, esfuerzo, constancia y dedicación todo es posible.

XVI. REFERENCIAS

[1] P. Morreale y J. Anderson, Software Defined Networking Design and Deployment, CRC Press, 2015.

[2] Open Networking Foundation, «Software-Defined Networking (SDN) Definition,» 2017. [En línea]. Available: https://www.opennetworking.org/sdn-definition/. [Último acceso: Noviembre 2017].

[3] P. Goransson y C. Black, Software Defined Networks A Comprehensive Approach, 2014.

[4] SDxCentral, «2017 Network Virtualization Report SDN Controllers, Cloud Networking and More,» 2017. [En línea]. Available: https://www.sdxcentral.com/reports/network-virtualization-sdn-controllers-download-2017/SDxCentral-Network-Virtualization-Report-2017-Rev-A.pdf.

[5] S. Azodolmolky, Software Defined Networking with Openflow, Birmingham: Packt Publishing, 2013.

[6] T. Nadeau y K. Gray, Software Defined Networks, O'REILLY, 2013.

[7] HP Networking, «HP Virtual Application Networks SDN Controller,» Noviembre 2017. [En línea]. Available: http://h17007.www1.hpe.com/es/es/solutions/technology/openflow/index.aspx. [Último acceso: Noviembre 2017].

[8] OpenStack, «Networking architecture,» Diciembre 2017. [En línea]. Available: https://docs.openstack.org/security-guide/networking/architecture.html.

[9] J. Donovan y K. Prabhu, Building the Network of the Future, CRC Press, 2017.

[10] OpenDaylight Project, *OpenDaylight Installation Guide*, 2016.

[11] J. Bermeo, M. Sanchez, J. Maldonado y J. P. Carvallo, «Modelos de Calidad de Software en la Práctica: Mejorando su Construcción con el Soporte de Modelos Conceptuales,» 2005.

[12] SDxCentral, «What is Ryu Controller?,» 2017. [En línea]. Available: https://www.sdxcentral.com/sdn/definitions/sdn-controllers/open-source-sdn-controllers/what-is-ryu-controller/.

[13] ONOS, «ONOS System Components,» Septiembre 2016. [En línea]. Available: https://wiki.onosproject.org/display/ONOS/System+Components.

[14] M. Noll, «Applied Research. Big Data. Distributed Systems. Open Source.,» Julio 2011. [En línea]. Available: http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/.

[15] CISCO, «Cisco Open SDN Controller,» 2017. [En línea]. Available: https://www.cisco.com/c/en/us/products/cloud-systems-management/open-sdn-controller/index.html.

[16] CISCO, «Event-Based Software-Defined Networking: Build a Secure Science DMZ,» California, 2015.

[17] OpenDaylight Project, «OpenDaylight Controller:Architectural Framework,» Marzo 2013. [En línea]. Available: https://wiki.opendaylight.org/view/OpenDaylight_Controller:Architectural_Framework.

[18] Linux Foundation Administrators, «What is ONOS?,» Septiembre 2017. [En línea]. Available: https://wiki.onosproject.org.

SOBRE LOS AUTORES

C**arlos Sandoval** nació en Cayambe en la provincia de Pichincha, el 08 de Febrero de 1992. Realizo sus estudios secundarios en el Instituto Tecnológico "Bolívar", obteniendo su título de Bachiller en Física y Matemática en el año 2009.
Es egresado de la Carrera de Ingeniería en Electrónica y Redes de Comunicación de la Facultad de Ingeniería en Ciencias Aplicadas en la Universidad Técnica del Norte de la ciudad de Ibarra.
Actualmente se desempeña como Gerente Técnico en la empresa Engine Digital Telecom en la ciudad de Cayambe.
Entre sus campos de interés están las Tecnologías de Comunicación de nueva Generación, el Diagnóstico de sistemas de Seguridad para redes cableadas e Inalámbricas y la administración de servidores bajo software libre.

Carlos A. VÁSQUEZ A. Nació en Quito - Ecuador el 19 de Septiembre de 1981. Ingeniero en Electrónica y Telecomunicaciones, Escuela Politécnica Nacional en 2008. Actualmente es docente de la Carrera de Ingeniería en Electrónica y Redes de Comunicación en la Universidad Técnica del Norte, Ibarra-Ecuador, posee además una Maestría en Redes de Comunicación, de la Pontificia Universidad Católica del Ecuador, Quito- Ecuador.

Implementation of an SDN cluster-controller based on a free software Framework for the Cloud Infrastructure of the Faculty of Engineering in Applied Sciences.

Carlos E.Sandoval
Faculty of Engineering in Applied Sciences
North Technical University
Ibarra, Ecuador
cesandovalc@utn.edu.ec

Carlos A. Vásquez
Faculty of Engineering in Applied Sciences
North Technical University
Ibarra, Ecuador
cavasquez@utn.edu.ec

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..

Palabras Clave—sdn; openflow; nfv; openvswitch; controller; cluster; agent; network; neutron; cloud; opensource; linux

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.

Keywords—sdn; openflow; nfv; openvswitch; controller; cluster; agent; network; neutron; cloud; opensource; linux

I. INTRODUCTION

Technologies have evolved so fast that they have turned networks into the point of failure, preventing them from growing and as a result they no longer satisfy the technological needs of companies, operators and users, which leads us to pursue a change in architecture that gives us flexibility, scalability, automation and control. [1]

Currently the requirements of organizations include shortening the physical space of the servers so it is recommended to work with network services also on the cloud. This demands certain sublayer requirements that can be covered by deploying an SDN over clustered nodes; Thus, these demands greatly challenge the capabilities they can provide and the traffic that common network devices are capable of supporting.

The Faculty of Engineering in Applied Sciences of the Technical University of the North in its Data Center has implemented a private cloud, which is limited in terms of the load that is able to support the infrastructure individually which does not represent an optimal scenario, reason by which it seeks to shorten spaces in order to benefit research projects on new network architectures in the faculty that allow in this way to cross the barrier established by traditional networks. Due to this, application servers have been implemented through cloud computing solutions of the IaaS type, but in a dispersed manner, which does not favor the management and administration of network services and capabilities.

Optimizing the physical space of the Data Center and reducing the carbon footprint is one of the most important objectives in the process of migration to new technologies, since this means greatly reducing energy consumption and emphasizing the strategic advantage of green design and construction.

II. SOFTWARE DEFINED NETWORKS

The Open Networking Foundation (ONF) is the organization most related to work on SDN. According to the ONF, Software Defined Networking is a more flexible architecture ideal for current applications that demand bandwidth dynamically. This architecture emancipates the functions of control and forwarding of data in the network allowing the network to be directly programmed, being dissociated from the underlying hardware for applications and services. OpenFlow becomes the main component for the manufacture of solutions based on Software Defined Networks in an agile, centralized way and based on open and neutral standards. [2]

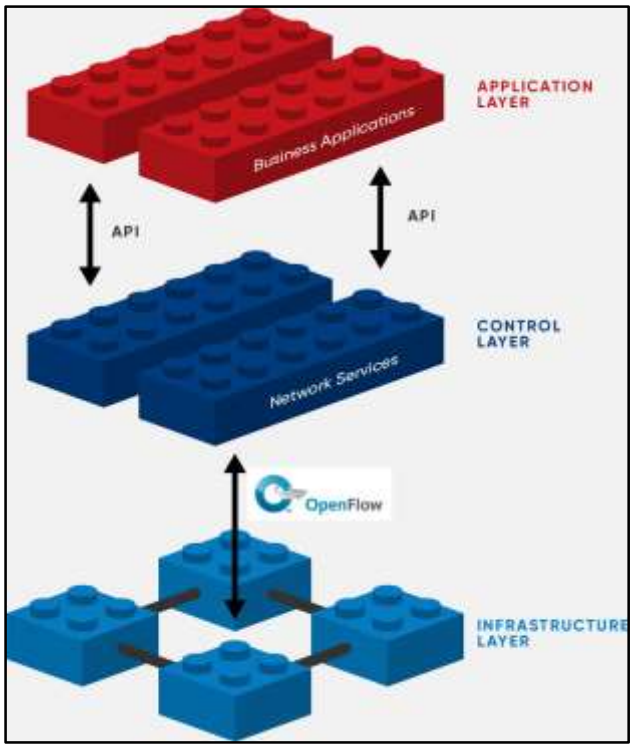

Fig. 1. Architecture and SDN elements

III. ARCHITECTURE AND ELEMENTS OF SDN

The devices that interconnect a network are composed of a data plane responsible for transmitting each packet through the network and a control plane that determines which is the best way for the information to reach its destination. Because SDN separates or abstracts these two planes it is possible to have an open interface to offer a solution to the demand and requirements of each connection point.

The general representation of the SDN is done through a diagram that shows the relationship of the two planes and their three embedded layers that are application, control and infrastructure of which the first two are managed in the control plane and the last is part of the data plane. The OpenFlow protocol acts as a connector between the infrastructure layer and the two higher ones through its communication with the NOS.

A. CONTROL PLANE

The control plane establishes the local data group used to create the entries of the forwarding table, which are used by the data plane to forward traffic between the access ports of a network device; this set of data representing a stored topology is called the Routing Table or Routing Information Base (RIB).

B. DATA PLANE

The data plane is responsible for handling incoming packets through a series of links regardless of the technology of the physical interface, during the collection process a series of consistency checks are made on the packages so that they can be sent then to the matching with the entries of the forwarding table programmed by the control plane.

C. NFV

Within the scope of SDN there is a very important term called NFV or Virtualization of Network Functions, around which there are confusions in the industry about whether or not it is a form of SDN. NFV is the implementation of the functionality of network devices and applications, but based on software. This means implementing a general component of the network as a load balancer or a software-based IDS. For this reason it is possible to virtualize routers and switches, thanks to advances in general purpose server hardware. [3]

IV. OPENFLOW

OpenFlow is the protocol that structures the communication between the control and data planes in the network devices that support it. It has been designed to provide an external application with access to the forwarding plane of a network switch or, failing that, of a router. Access to this part of the router can be obtained through the network, which allows the control program does not have to be located near the switch. [4]

Due to the importance that was expected with the change to SDN, the participating organizations of the OpenFlow project encouraged the use of this specification in multiple test environments in order to establish standards and achieve the implementation of the SDNs beyond the academies and research groups. experimental development. From this momentum OpenFlow managed to stand out and began to be included in a wide variety of firmwares and network operating systems of the member organizations of the project, achieving a quite high degree of evolution in a short time. The existing functional versions of the OpenFlow specification range from 1.0 to 1.5, of which 1.4 and 1.5 are still in development and are intended to include support for fiber optic networks natively, as well as batches of extended instructions for the flow tables. [5]

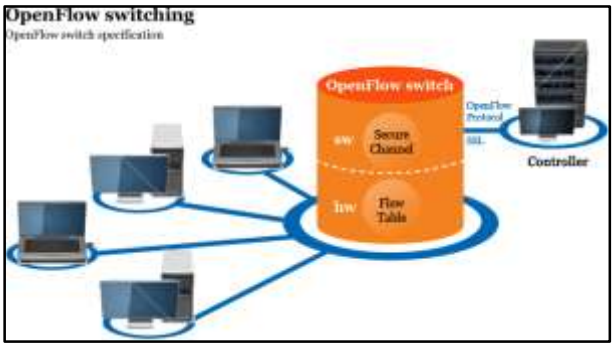

Fig. 2. Openflow Switching.

V. SDN CONTROLLERS

The general description of an SDN controller is that of a system based on software or set of systems that in contrast can provide a series of services or capabilities that allow specific tasks to be carried out in the devices that are part of the software defined network and in some cases integrate those outside the OpenFlow communication channel. Among the functions provided by a controller are:

- Management of the state of the network, and in some cases, the management and distribution of status through databases.
- High-level data model that captures relationships between managed resources, policies and other services provided by the controller using the Yang modeling language.
- Modern application programming interface (API), often RESTful that publishes the services of the controller and application.
- Secure control session via TCP between the controller and the associated agents in the network elements.
- Standards-based protocol for the provisioning of network status driven by applications in network elements.
- Mechanism for device and topology discovery, route calculation system and potentially other information services centered on the network or focused on resources.

Currently, there are different implementations of OpenFlow and SDN controllers, most are built on open source resources and others have been developed by major hardware manufacturers such as Cisco, Big Switch Networks and VMware. Some OpenFlow controllers present features that others do not have and are written in different languages, so the selection of one of them must be done according to the type of implementation that is being sought. [4] [7]

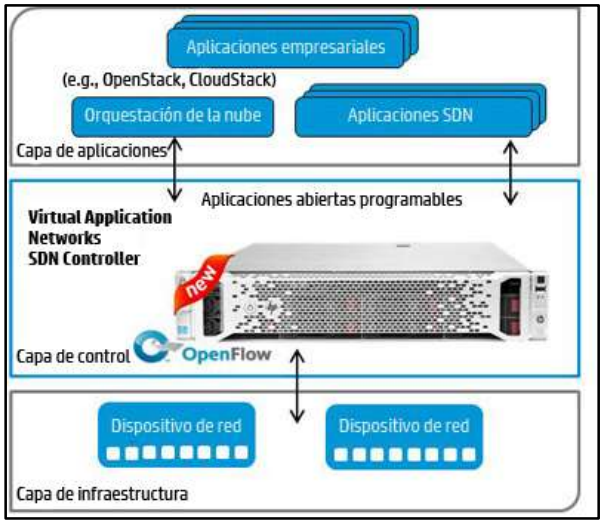

Fig. 3. HP SDN Controller

VI. CLOUD COMPUTING NETWORKS

The decrease in the importance of transport and the emergence of a fully IP network platform with demand for various services creates opportunities, or in some cases a requirement for telecommunications providers to remain relevant when adopting a new framework for the provision of services. The essential elements of the new framework include hardware elements for reengineering software functions running on the cloud computing infrastructure. [8]

Alignment of access networks in both core and edge using traffic patterns created by IP-based services; the integration of network and cloud technologies in a software platform that allows rapid implementation, administration of services with high automation; a software-defined control so that both the infrastructure and the functions can be optimized by changing the demand for the service and the availability of infrastructure; an increase in software integration competencies and a DevOps operations model. This is known as "Network Cloud". [5]

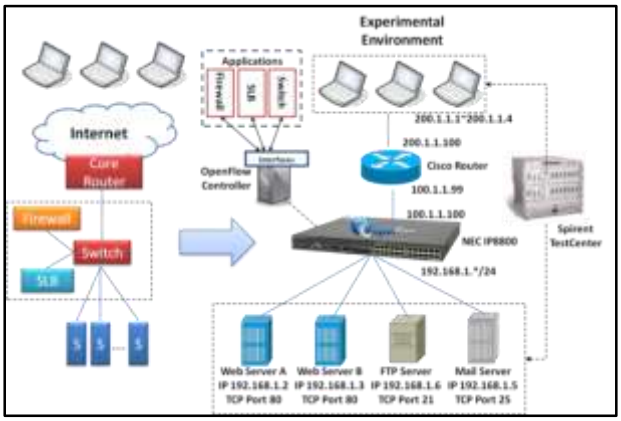

Fig. 4. Simplified architecture of a cloud computing network.

VII. CLUSTERING

Clustering is a mechanism that allows multiple processes and programs to work together as a single entity. When searching the world wide web, it may appear that the search request is processed by a single web server. In reality, the search request is processed by a large group of clustered web servers. In the same way, you can have several instances of the same service working together as a single entity to obtain certain advantages such as:

- Scalability: if you have several instances of running a service, you can do more work and store more data than you could with a single instance. You can also divide the data into smaller pieces and distribute them in the cluster or perform certain operations on certain members of the cluster.
- High availability: If there are several instances of execution of a service and one of them suffers a failure, the other instances will remain functional and available.
- Data Persistence: No data stored on the servers will be lost after a manual, scheduled restart or a lock. [6]

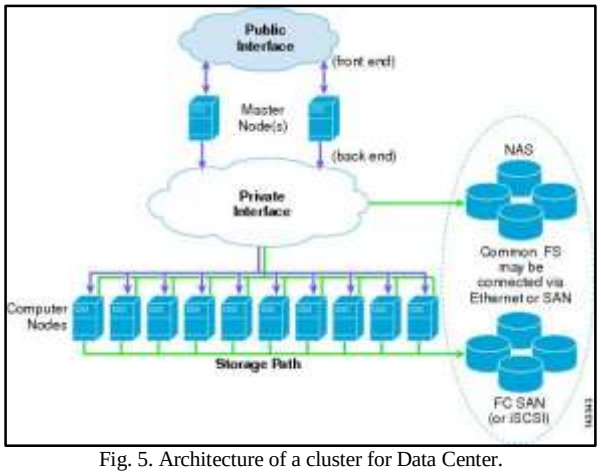

Fig. 5. Architecture of a cluster for Data Center.

VIII. CURRENT SITUATION DATACENTER FICA

The Faculty of Engineering in Applied Sciences has a fully functional TIER I data center, in charge of hosting a series of local services. The structure of the network of the Universidad Técnica del Norte and the FICA data center is shown in the following figure, where you can see what is the current distribution and their respective labels.

In the area corresponding to the equipment of the FICA data center, a distribution switch is located, which is in turn connected to another non-administrable switch of the Linksys brand, which serves as a bridge to the servers of the cloud section and to the equipment They provide local services such as Opinion, Reagents, Geoportal and DSpace. Each of these devices is assigned an IP address belonging to the pool of the demilitarized zone (DMZ) of the Universidad Técnica del Norte and also has a local access address generated from the address pool of the institution's private network.

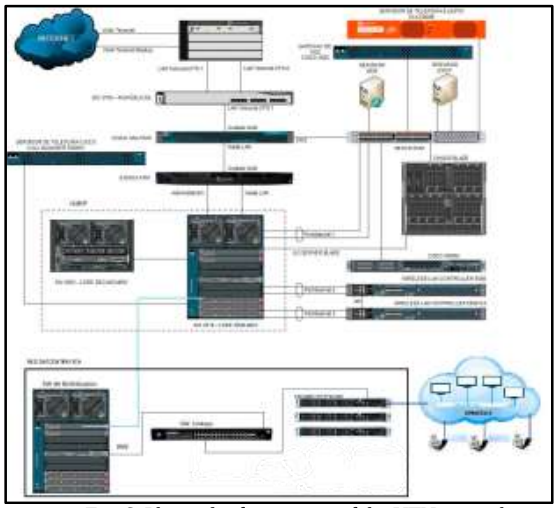
Fig. 6. Physical infrastructure of the UTN network

The deployment of SDN integrated into the cloud requires that the characteristics of the cloud platform be consciously evaluated so that the result is satisfactory. That is why a process for the creation of an open source platform evaluation matrix is executed, based on the steps of the IQMC method which provides the guides and techniques to identify the appropriate quality factors of the software components. [7]

By applying this method the result must comply with four principles:

- Must have high quality quality characteristics.
- Must allow hierarchy in the quality characteristics to structure the model.
- Must allow concealment to avoid omitting dependent characteristics.
- Must be general to avoid features not useful for software engineering.

After constructing the evaluation model and generating the matrices with the parameters established by the ISO / IEC 25000 standard, the metrics are applied to the attributes and the sum of the scores assigned to each of the quality characteristics is performed; obtaining the results that are detailed in the Table. The evaluated percentage of each characteristic is obtained based on the parameters of the Quality Assessment Division that are described in ISO / IEC 2504n.

TABLA I
Resultados del análisis de calidad del software según la norma ISO/IEC/IEEE 2504n

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

IX. CONDITIONING THE PLATFORM

Due to the modular nature of OpenStack, the integration of additional elements requires that adaptations or configurations are made in the service daemons so that the correct operation is not affected after adding agents from third parties. The advantage that OpenStack has is its intuitive combination method based on libraries written for specific purposes. [8]

This stage consists in the reconstruction of the cloud platform so that it meets the needs of SDN. This is achieved by reinstalling and reconfiguring the compute and network modules to work in parallel with the network control node that is deployed in a subsequent section. We must ensure that the

operating system is fully updated to obtain the most recent infrastructure manager packages.

The first thing to do is download the most current OpenStack repositories and install the PackStack tool as shown in Figure 60. Once this has been done, proceed to generate the response file as shown in Figure 61 and, a from this file the environment to be deployed is created, modifying it according to the type of use that will be given to the platform.

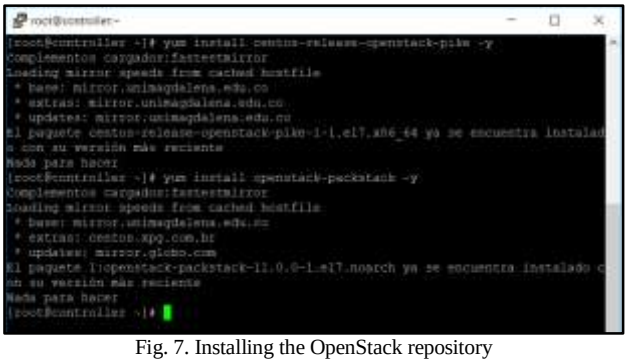

Fig. 7. Installing the OpenStack repository

The necessary and most important fields are modified to proceed to execute the deployment using the PackStack tool which will take some time to elaborate the platform depending on the processing capacity of the server and the internet connection to download the necessary packages from the OpenStack repository.

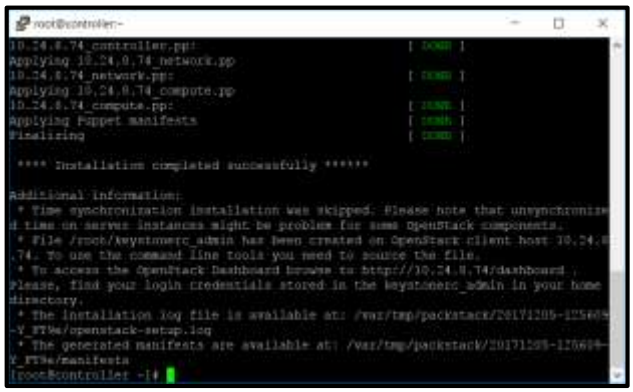

Fig. 8. Successful installation of OpenStack.

To finish with the conditioning of the cloud platform, the status of the services Keystone (Identity), Glance (Image Storage) and Nova (Compute) is verified, through the creation of a flavor, a network and a test instance through the manager of infrastructure.

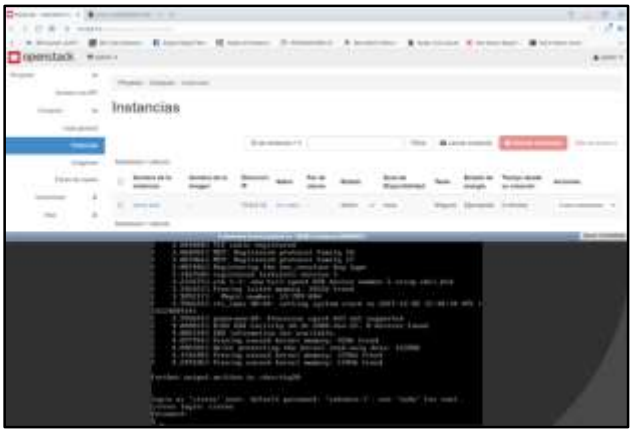

Fig. 9. Instance running on OpenStack.

In this section, the Neutron module is also configured to perform the implementation of a classic OpenStack Networking service scenario using the ML2 plug-in with Open vSwitch. This implementation provides a method for non-privileged users to access virtual networks within a project and includes the following components:

- Project networks (tenant): provide connectivity to instances for a particular project. Users without privileges can manage project networks within the assignment assigned by the administrator to them. Project networks can use the VLAN, GRE or VXLAN transport methods.
- External networks: provide connectivity to external networks such as the internet. Only administrative users can manage external networks because they interact with the physical network infrastructure. External networks can use flat or VLAN transport methods, depending on the physical infrastructure of the network and generally use ranges of public IP addresses.

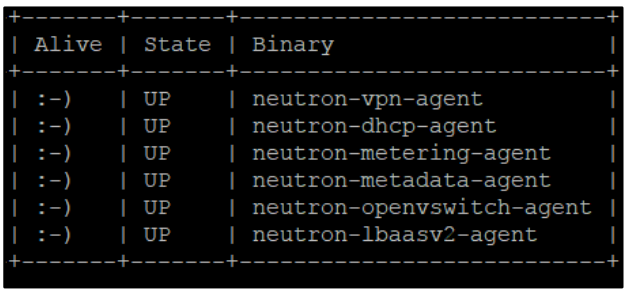

Fig. 10. Agents of the Neutron network module.

X. SDN CONTROLLERS EVALUATION

There are multiple solutions for SDN network controllers, both open source and paid. One of the objectives of this project is to provide a free solution, through the use of software-based solutions that adapt to the hardware that is currently available.

For the evaluation of software-based controllers, an experimental environment based on virtual machines is used on the VMware Workstation hypervisor. In this section we analyze

the functionalities of each controller using a very useful tool for network simulation called mininet. A clip art is downloaded to focus directly on the functional features of the control software. Two virtual machines are used. In the first one, the controllers are installed one by one for the respective tests, and in the second, the Mininet network simulator is installed, which will be responsible for providing the network devices that will appear in the controller once both services have been integrated.

Not all existing controllers are candidates for evaluation, since several are discontinued or do not fall into the software-based group. The controllers that will undergo evaluation for the selection of the controller that will be implemented in the SDN network will be:

- RYU
- FLOOFLIGHT
- OPENDAYLIGHT
- ONOS

A. RYU TEST

Once the RYU software has been downloaded it is necessary to install certain extra dependencies that are not exposed in the official documentation. This happens because of the little support these packages receive due to their usability in production environments. When all dependencies have been satisfied, the driver software is executed from the command line.

With the controller in functional status, it is allowed to use the modules that are offered separately. To carry out the test, the simple switching module is used together with a simulated network from the mininet simulation software hosted on another equipment. First we will execute the instruction to initialize the controller, and then we start the simulation to appreciate the instant notification messages generated by RYU. [12]

Fig. 11. Communication established between mininet and RYU.

The second part of the test is to make a brief packet capture to check how the OpenFlow session is carried out and what type of packets are being transmitted. For this the wireshark tool is used, installed on the same server where the control software is hosted.

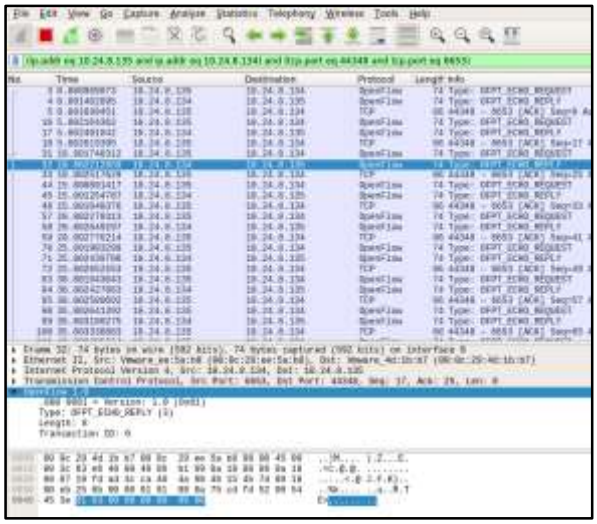

Fig. 12. Traffic capture OpenFlow between RYU and mininet.

B. FLOODLIGHT TEST

For the evaluation of features and functionalities of the Floodlight controller, a virtual machine is used on the VMware Workstation hypervisor, in which the operating system Ubuntu 16.04.3 LTS has been installed. To start it is necessary to download the Linux development toolset and the git and ant utilities. This with the aim of compiling the source code of the Floodlight project.

Fig. 13. Floodlight user interface.

A basic test can be performed using the mininet network simulator. This simulator works based on execution scripts or parameterized commands.

Fig. 14. Creation of a test OpenFlow network.

The second test is technical, and will be done by using the wireshark packet capture tool. In fact, it is not necessary to have a graphical environment on the server that acts as a controller since there are also tools for capturing and analyzing packages in command line version. For experimental purposes, the distribution of Ubuntu in its graphic version has been used.

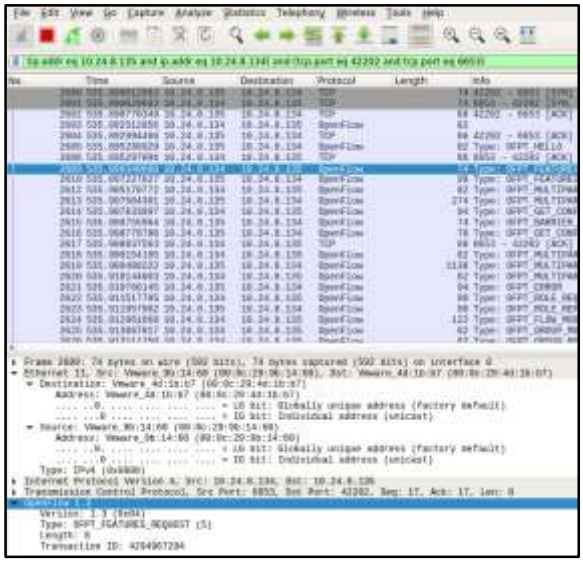

Fig. 15. Traffic capture over the Floodlight controller interface.

C. PRUEBA CON OPENDAYLIGHT

OpenDaylight is written in java so it requires certain specific components to work, among these is the Oracle Java development kit. Before starting, we must ensure that all dependencies are satisfied and there is no compilation error in the program. Once this is done we start the software and enter the controller programming console.

Initially OpenDaylight does not have any module installed since it has a wide collection of components. For this reason we proceed to add the basic elements for the creation of the test environment. With the already loaded modules, the controller acquires new capacities among them the creation of dynamic flow inputs, creation of automatic flow tables and generation of the graphic topology. [10]

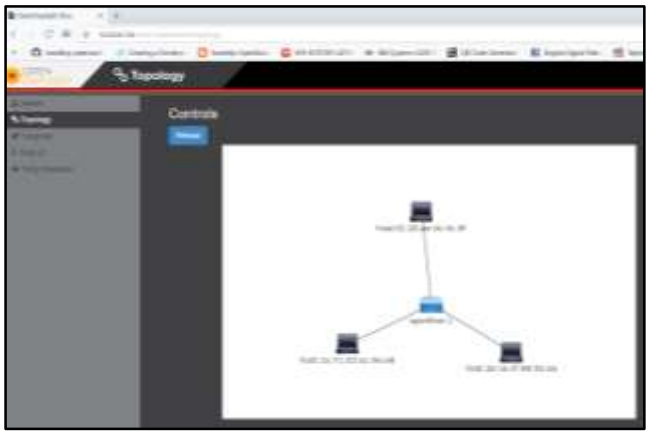

Fig. 16. OpenDaylight DLUX interface.

Once the controller and its modules are active, traffic is generated between the interfaces of the simulated network devices and the OpenDaylight controller. Instantly, the flow tables are updated and show the number of transmitted packets. To see more deeply the status of the session, the messages are filtered to observe only those belonging to the OpenFlow conversation.

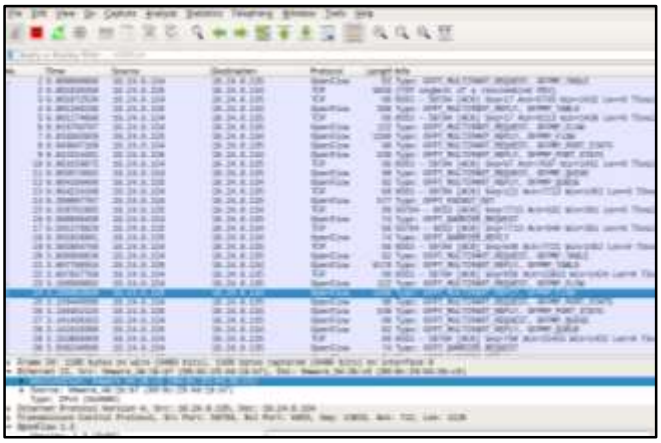

Fig. 17. OpenDay session of OpenDaylight in Wireshark

D. ONOS TEST

For the tests of functionality of Open Network Operating System we will adapt an environment with two virtual machines in which we will install the software of the controller and the tool mininet. ONOS is a platform written in java and uses OSGi to manage its functions as well as OpenDaylight. Its characteristics are loaded using its execution time called Karaf.

Once the basic modules are activated, from the virtual machine with mininet, a test topology is generated by connecting one or more switches to the controller, pointing the simulator to the ONOS IP. Once we execute the simulation instruction in mininet, from the ONOS console we verify that the controller learned new information from the connected hosts. [13]

Fig. 18. Discovery of hosts in ONOS Controller

Once the functionality of the controller has been successfully verified, the analysis of the packages immersed in the OpenFlow session between ONOS and the network elements simulated in mininet is carried out. This analysis is developed using the wireshark software, in which we use the physical network interface of the server that hosts the network controller to capture the traffic generated in the communication through OpenFlow.

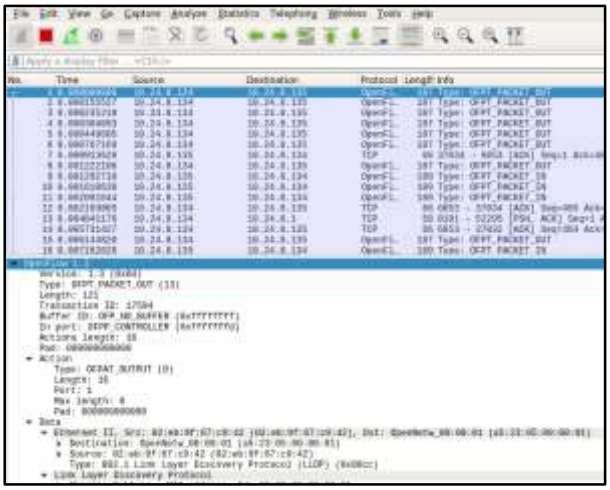

Fig. 19. PACKET_OUT message generated by ONOS to the simulated switch.

XI. SELECTING THE CONTROLLER SOFTWARE

The selection of the driver software that will be used for the deployment is not done arbitrarily. For this task it is necessary to use a selection method that allows choosing the best option according to the requirements of the project. In this case, a selection model will be used based on the parameters imposed by the ISO / IEC / IEEE 29148 standard in its amendment of the year 2011. This standard presents the requirements and parameters for the selection of software for an implementation project. These parameters are specified in the section "9.5 Specification of Software Requirements (SRS)".

Generally, the software selection process obeys the need to establish selection criteria within the pre-development activities. These criteria comprise a set of parameters closely linked to the needs of the users, and serve as reference for the evaluation and, therefore, are the key to the selection decision.

- Purpose
- Reach
- Product perspective
- User characteristics
- Functional requirements
- Non-functional requirements
- Performance requirements

According to the results of the evaluation lists for software selection, the one that complies with all the requirements established by ISO / IEC / IEEE 29148 is OpenDaylight. This presents the best features and features, so it is perfectly suited to the deployment proposed in this project.

OpenDaylight has the advantage of being compatible with physical and virtual switches, presents a friendly web interface and has support for API REST. Another important feature is its compatibility with more than one operating system and the most current versions of java. Its installation is simple, its development is very active and also fulfills the functional and non-functional requirements completely. This makes it a very robust software which guarantees optimum and efficient performance covering the needs of the SDN infrastructure.

XII. SDN CONTROLLERD DEPLOYMENT

The OpenDaylight version used during this deployment is Nitrogen, which is the most current to date. To obtain a copy of the software, we go to the ODL project website indicated in Figure 85. In the Downloads section we will find the packages and their respective documentation hosted on a wiki with the most recent distribution information, such as the version file. previous of the distribution.

In case any step has been omitted during the hardware start-up, it is corrected immediately to prevent any unintentional error. The server parameters must be in the same communication group of the cloud servers and must be set correctly

After installing the Java development package, it is necessary to go to the directory where the OpenDaylight package is located to extract the content. When the extraction finishes, a folder containing all the files that make the driver software work will be created. At this point it is possible to see that there are other directories with executables for different purposes, but the one that should be used to start ODL is the one with the name karaf inside the folder / bin.

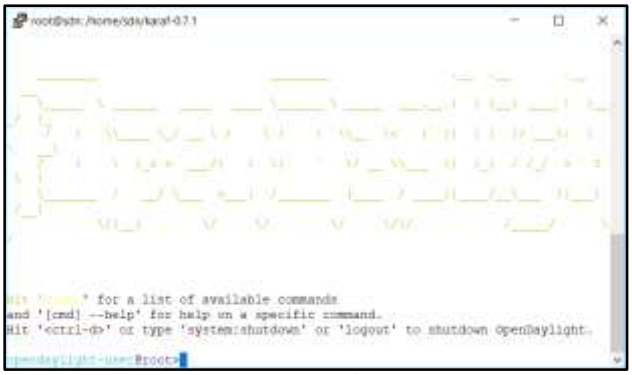

Fig. 20. OpenDaylight configuration console.

Because the deployment consists of the use of a network defined by hybrid software, it is necessary to enable the layer two forwarding module, the application nodes for the OpenFlow integration and the DLUX graphical interface.

The next step to start the controller is to make sure that the most recent version of Openflow is used, for this it is convenient to stop Karaf while the configuration is done. OpenDaylight incorporates in its latest versions the ability to support both version 1.0 and 1.3. Because the Open vSwitch software used as a Back-end for the cloud environment network has the option of communications enabled by OpenFlow 1.3, you must configure the custom.properties file located in the <odl folder> / etc / directory, in the line ovsdb.of.version = 1.3 which ensures that there are no inconsistencies in the session messages between ODL and the nodes belonging to the virtualization environment. The next thing is to restart Karaf and wait for the configuration

to be reloaded in order to enter the DLUX interface with the new loaded components.

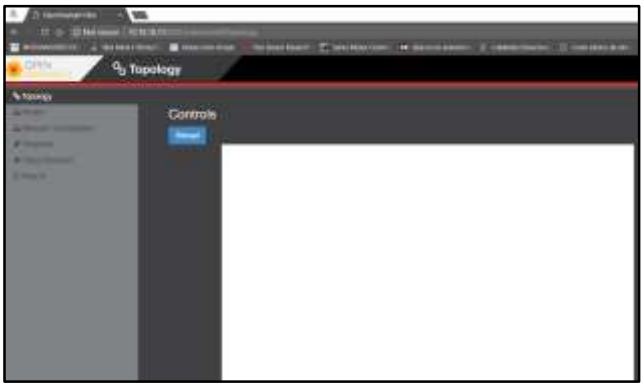

Fig. 21. Graphical environment of the NetVirt and DLUX modules

For the installation process of the new network agent, it is necessary to follow a series of steps in a certain order since otherwise there would be inconsistencies in the OpenStack services that depend on Neutron and its related components. First, it is imperative to ensure that using OpenDaylight as the Neutron back-end, ODL is the only real source for Neutron configurations. Due to this it is necessary to eliminate the existing OpenStack configurations to provide OpenDaylight with a clean platform on which to work using the instructions provided by the infrastructure manager on the command line.

In order to verify that the integration has been carried out successfully, an ssh session must be opened in the main OpenStack server and after obtaining privileges with the key generated by Keystone, the instruction to list the network agents is executed, where now appears the Layer 2 agent belonging to OpenDaylight. This confirms that the ODL agent is integrated with OpenStack.

```
enstack network agent list
+--------------------+------------+-------------------+-------+-------+
| Agent Type         | Host       | Availability Zone | Alive | State |
+--------------------+------------+-------------------+-------+-------+
| ODL L2             | controller | None              | True  | UP    |
| L3 agent           | controller | nova              | True  | UP    |
| DHCP agent         | controller | nova              | True  | UP    |
| Metadata agent     | controller | None              | True  | UP    |
+--------------------+------------+-------------------+-------+-------+
```

Fig. 22. Network agent ODL L2

XIII. FUNCIONALITY TEST

When OpenDaylight is integrated with OpenStack, the switches and connectors of each virtual switch are displayed in the Nodes tab. In the case of the platform controller, initially it shows a single connector called br-int. This is the bridge that is responsible for interconnecting all virtualized devices in OpenStack using the resources stored in the OpenDaylight datastore and the flow tables predetermined by NetVirt. In this way, the Openflow communication channel from East to West, that is, horizontal traffic, is mechanically constructed.

Fig.23. Flow tables reflected in ODL.

The last stage of the implementation is the clustering of the two remaining nodes with the same characteristics of the control node. This process consists of two parts, the first is the provisioning from the main node by injecting the configuration scripts to the slave nodes and the second reconfiguration of the files related to the Neutron service so that they are linked to the SDN controller using the OVSDB interface of the NetVirt platform.

At this point all the nodes have the virtual switch enabled but they do not maintain communication with each other and they are not managed through OpenFlow. To enable the Open vSwitch of each node to work under the control of OpenDaylight, it is necessary to edit the Neutron network agent files in a similar way to the main node.

Immediately after the controller registers the nodes in the datastore, connectors are created for each virtual switch with a unique ID that is used to build the logical links in the NetVirt platform, shortly after these links are generated the topology is represented to through the Network Virtualization tab in the DLUX environment.

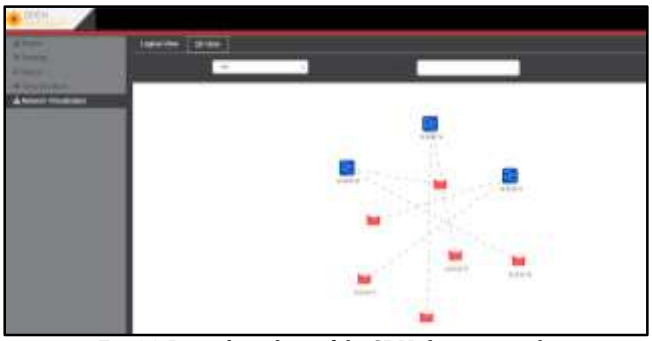

Fig. 24. Logical topology of the SDN cluster network.

The communication between the nodes uses the tunneling method for networks overlay, which allows the logical separation of the data that circulates through the cloud infrastructure. It is for this reason that if a traffic capture is made directly on the physical interface of the computing node, packages with illegible information will appear. To be able to analyze a package with encapsulation it is necessary to use a dissector or a packet decoder like the one provided by Wireshark.

To visualize the content and fields of a package with VXLAN encapsulation it is possible to use the tshark tool on the command line to generate a traffic record that can then be read more carefully in Wireshark through the graphical interface. The instruction that is used to generate this record has a filter for the default VXLAN port that is 4789.

It can be seen that the ICMP packets are already decoded, this is because the version of Wireshark for Windows already incorporates the Openflow dissector. Among the captured

packets you can see the source and destination IP addresses of the instances that are inside OpenStack, as well as the source and destination ports of the VXLAN tunnel.

To verify that the information provided is correct, you can check the OpenDaylight datastore to verify the tunnel's VNI, which in this case is 68. To do this, use the API interface to consult the networks section and compare if the value is the same.

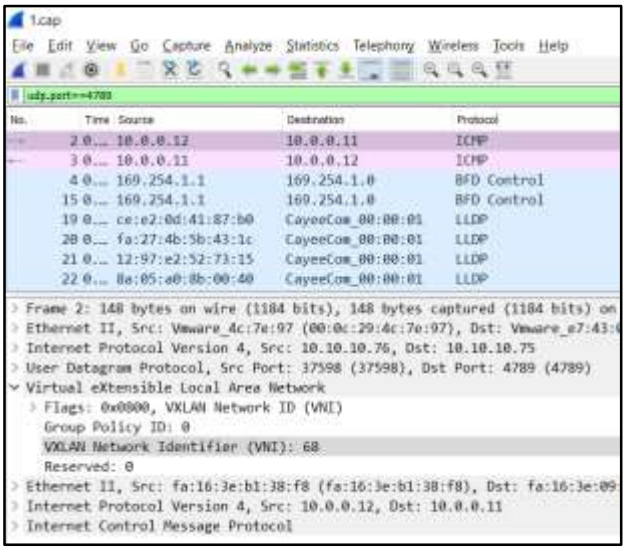

Fig. 25. VXLAN traffic decoded in Wireshark.

The internal tunnel manager creates and maintains the mesh of VXLAN or GRE type tunnels, among the Openflow switches that form an overlay transport network. ITM has two modes of operation, the first one creates the tunnels under command, that is automatically and this is the option enabled by default. The second mode is manual and allows you to enter pre-configured tunnels via REST_API. When configured in this way, it is important to establish the transport zones correctly, otherwise rules may be missing and the connection will fail .

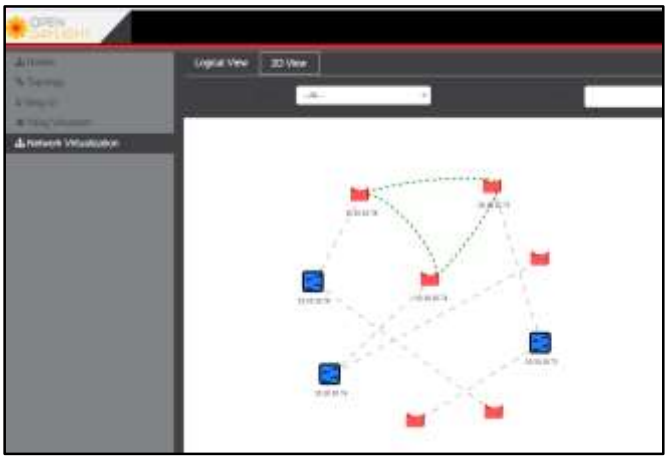
Fig. 26. Ring of VXLAN tunnels created by ITM.

XIV. CONCLUSIONS

The implementation of the software-defined network for the FICA Data Center was based on the open source virtualization elements and tools provided by OpenStack and Opendaylight in a satisfactory manner, having carried out the convergence tests necessary to confirm the validity of the deployment , as well as the culmination of the project.

Among the possible alternatives for the network controller, only those projects that during the development of this deployment are active and maintain support from their creators were subjected to evaluation and preliminary tests. The selection matrix showed that OpenDaylight is currently the most robust project and compatible with SDN and OpenStack platform modules.

After the successful implementation of the network controller, it was possible to understand the scope of platform integration, allowing to provide more robust services, more reliable and with a scalability index much greater than that provided using only service managers individually.

Traffic encapsulated with VXLAN over virtual networks represents an advantage over competing physical devices since the latter need some element to manage the encapsulation of the network to inject it into a different virtual network or the physical network. That is why the use of SDNs is increasingly becoming a success rather than a challenge.

XV. GRATITUDE

To the Technical University of the North and the Faculty of Engineering in Applied Sciences, to the entire teaching staff that for so long has prepared me to face the challenges of professional life.

A sincere and special thanks to my director, Eng. Carlos Vásquez for the guidance during my time in university life and for giving me the opportunity to prove that there is no impossible project and that, with hard work, effort, perseverance and dedication everything is possible.

XVI. REFERENCES

[1] P. Morreale y J. Anderson, Software Defined Networking Design and Deployment, CRC Press, 2015.

[2] Open Networking Foundation, «Software-Defined Networking (SDN) Definition,» 2017. [En línea]. Available: https://www.opennetworking.org/sdn-definition/. [Último acceso: Noviembre 2017].

[3] P. Goransson y C. Black, Software Defined Networks A Comprehensive Approach, 2014.

[4] SDxCentral, «2017 Network Virtualization Report SDN Controllers, Cloud Networking and More,» 2017. [En línea]. Available: https://www.sdxcentral.com/reports/network-virtualization-sdn-controllers-download-2017/SDxCentral-Network-Virtualization-Report-2017-Rev-A.pdf.

[5] S. Azodolmolky, Software Defined Networking with Openflow, Birmingham: Packt Publishing, 2013.

[6] T. Nadeau y K. Gray, Software Defined Networks, O'REILLY, 2013.

[7] HP Networking, «HP Virtual Application Networks SDN Controller,» Noviembre 2017. [En línea]. Available:

http://h17007.www1.hpe.com/es/es/solutions/technology/openflow/index.aspx. [Último acceso: Noviembre 2017].

[8] OpenStack, «Networking architecture,» Diciembre 2017. [En línea]. Available: https://docs.openstack.org/security-guide/networking/architecture.html.

[9] J. Donovan y K. Prabhu, Building the Network of the Future, CRC Press, 2017.

[10] OpenDaylight Project, *OpenDaylight Installation Guide*, 2016.

[11] J. Bermeo, M. Sanchez, J. Maldonado y J. P. Carvallo, «Modelos de Calidad de Software en la Práctica: Mejorando su Construcción con el Soporte de Modelos Conceptuales,» 2005.

[12] SDxCentral, «What is Ryu Controller?,» 2017. [En línea]. Available: https://www.sdxcentral.com/sdn/definitions/sdn-controllers/open-source-sdn-controllers/what-is-ryu-controller/.

[13] ONOS, «ONOS System Components,» Septiembre 2016. [En línea]. Available: https://wiki.onosproject.org/display/ONOS/System+Components.

[14] M. Noll, «Applied Research. Big Data. Distributed Systems. Open Source.,» Julio 2011. [En línea]. Available: http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/.

[15] CISCO, «Cisco Open SDN Controller,» 2017. [En línea]. Available: https://www.cisco.com/c/en/us/products/cloud-systems-management/open-sdn-controller/index.html.

[16] CISCO, «Event-Based Software-Defined Networking: Build a Secure Science DMZ,» California, 2015.

[17] OpenDaylight Project, «OpenDaylight Controller:Architectural Framework,» Marzo 2013. [En línea]. Available: https://wiki.opendaylight.org/view/OpenDaylight_Controller:Architectural_Framework.

[18] Linux Foundation Administrators, «What is ONOS?,» Septiembre 2017. [En línea]. Available: https://wiki.onosproject.org.

ABOUT AUTHORS

C**arlos Sandoval** was born in Cayambe in the province of Pichincha, on February 8, 1992. He completed his secondary studies at the Technological Institute "Bolivar", obtaining his Bachelor's degree in Physics and Mathematics in 2009.
He graduated from the Electronics Engineering and Communication Networks of the Faculty of Engineering in Applied Sciences at the Technical University of the North of the city of Ibarra. Currently he works as Technical Manager in the company Engine Digital Telecom in the city of Cayambe.
Among its fields of interest are the new generation of Communication Technologies, the Diagnosis of security systems for wired and wireless networks and the administration of servers under free software.

Carlos A. VÁSQUEZ A. was born in Quito - Ecuador on September 19, 1981. Electronics and Telecommunications Engineer, National Polytechnic School in 2008. He currently teaches the Electronics Engineering and Communication Networks at the Universidad Técnica del Norte, Ibarra-Ecuador, owns also a Master's Degree in Communication Networks, from the Pontifical Catholic University of Ecuador, Quito-Ecuador.