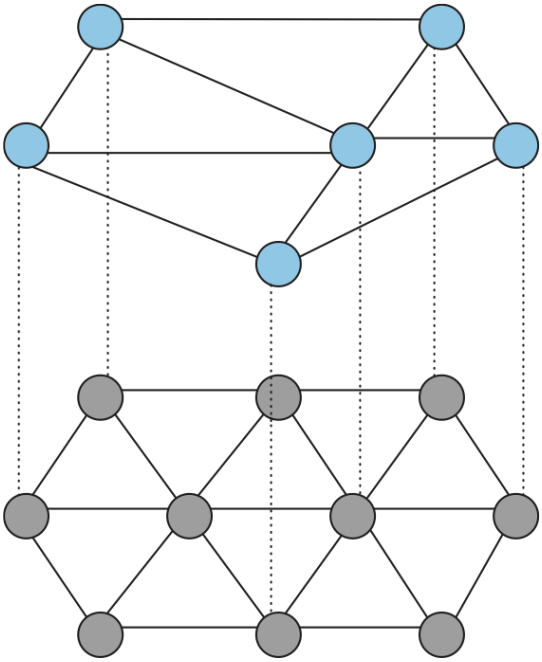


ESTUDIO E IMPLEMENTACIÓN DE TECNOLOGÍAS OVERLAY EN REDES DEFINIDAS POR SOFTWARE SDN

Bárbara Castro Serantes

Grado Universitario en Ingeniería Tecnologías y Servicios de Telecomunicación

Área: Administración de redes y Sistemas Operativos

Tutor: Joaquín López Sánchez-Montañés

15 de enero de 2023

No hay ninguna fuente en el documento actual.

GNU Free Documentation License (GNU FDL)

C) Copyright

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Estudio e implementación de tecnologías overlay en redes definidas por software (SDN)*
Nombre del autor:	*Bárbara Castro Serantes*
Nombre del consultor/a:	*Joaquín López Sánchez-Montañés*
Nombre del PRA:	*Montse Serra Vizern*
Fecha de entrega (mm/aaaa):	15/01/2023
Titulación:	*Grado universitario en Ingeniería Tecnologías y Servicios de Telecomunicación*
Área del Trabajo Final:	*Administración de redes y sistemas operativos*
Idioma del trabajo:	*Castellano*
Palabras clave	*SDN, overlay network, openflow, vxlan, geneve,onos*

Resumen del Trabajo:

La necesidad de desarrollo de las redes basadas en software o SDN, nacen bajo el paradigma de la computación en la nube o *Cloud Computing* y la virtualización. La complejidad física de la red actual y la gran cantidad de protocolos existentes en entornos en los que abundan los desarrollos propietarios o privados, dificultan la gestión eficiente de la red.

Actualmente, los arquitectos de red, ante la elevada demanda de datos, buscan soluciones más flexibles que les permitan gestionar de manera eficiente estos recursos.

En este TFG se estudiarán las redes definidas por software, sus tecnologías, protocolos, arquitecturas y mecanismos de control de tráfico. Se expondrán las limitaciones de las redes actuales y cómo SDN es capaz de solventar algunos de estos problemas mediante la integración de las ventajas que introduce este nuevo paradigma, en comparación con las tecnologías utilizadas hasta ahora en un entorno tanto teórico como práctico.

En cuanto al desarrollo del TFG, en primer lugar, se desplegará un laboratorio virtual con la demostración del funcionamiento de un controlador centralizado en SDN.

En segundo lugar, se analizarán las principales tecnologías aplicadas al entorno SDN y se realizarán pruebas dentro de un laboratorio virtual de desarrollo propio.

En ambos casos prácticos se dará prioridad a las soluciones *opensource* o de código abierto, siempre que sea posible.

En conclusión, se pretende defender las ventajas de SDN, por su escalabilidad, adaptabilidad y flexibilidad, generando en paralelo un adecuado punto de partida para nuevos estudios de mayor profundidad que pudieran ser de interés.

Abstract:

The need for the development of software-based networks or SDNs is born under the paradigm of cloud computing and virtualisation. The physical complexity of the current network and the large number of existing protocols in environments where proprietary or private developments abound, make efficient network management difficult.

Currently, network architects, faced with the high demand for data, are looking for more flexible solutions that allow them to efficiently manage these resources.

This dissertation will study software-defined networks, their technologies, protocols, architectures and traffic control mechanisms. The limitations of current networks and how SDN is able to solve some of these problems by integrating the advantages introduced by this new paradigm, compared to the technologies used until now, will be presented in a theoretical and practical environment.

Regarding the development of the TFG, firstly, a virtual laboratory will be deployed with the demonstration of the operation of a centralised controller in SDN.

Secondly, the main technologies applied to the SDN environment will be analysed and tests will be carried out in a virtual laboratory of our own development.

In both case studies, priority will be given to open source solutions, whenever possible.

In conclusion, the aim is to defend the advantages of SDN, due to its scalability, adaptability and flexibility, generating in parallel a suitable starting point for new studies of greater depth that could be of interest.

INDICE

TABLAS

ILUSTRACIONES

1. Introducción

1.1 Contexto y justificación del Trabajo

En la última década, el aumento del volumen de datos en los centros de procesado de datos se ha incrementado considerablemente dejando al descubierto las limitaciones de las arquitecturas tradicionales y sus tecnologías.

Desde hace unos años, las redes de datos están experimentando su tercera revolución. La primera revolución, fue el paso de conmutación de circuitos a conmutación de paquetes, la segunda revolución, fue el paso de medios cableados a medios inalámbricos y, finalmente, la tercera revolución, que se analizará en este proyecto, será el paso de un modelo de infraestructuras basadas en hardware a un modelo de infraestructuras basadas en software, *Software Defined Network* (SDN).

La computación en la nube o *Cloud Computing* está impulsando la transición de las empresas hacia las redes definidas por software para hacer frente al consumo frenético de datos por parte de los usuarios.

Redes sociales, tales como Instagram, Facebook, Twitter, plataformas de contenido, tales como Netflix, Amazon Prime, HBO, Disney+ o plataformas de Streaming, tales como Twitch, YouTube, han visto como triplicaban el consumo de datos a volúmenes nunca vistos.

Según un estudio estadístico por parte de la empresa Statia, el volumen de tráfico de red para entrega de contenidos o *Content Delivery Network* (CDN) desde el 2017 al 2022 ha superado los 258.000 Petabytes mensuales [1]. Y se espera que en el año 2025 el volumen de datos podría llegar a los 175 zettabytes. [2]

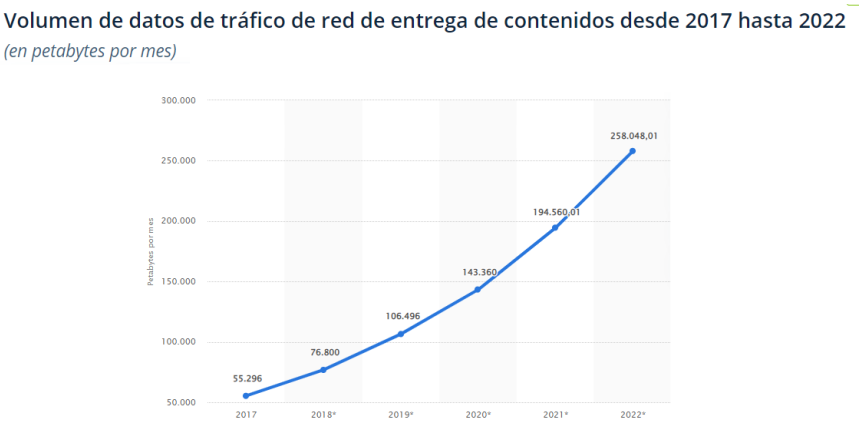


Ilustración 1 - Volumen de tráfico de datos 2007 a 2022

La situación actual de Internet ha potenciado el auge de SDN y las principales compañías y operadoras del sector tecnológico, tales como Amazon, Nokia, Ericsson, Fortinet, Cisco, VMware, Juniper, Huawei o el propio *Metro Ethernet Forum (MEF)* [3] han creado una división específica dentro de su organización para el estudio de tecnologías SDN. Del mismo modo, institutos y fundaciones como *Institute of*

*Electrical and Electronics Engineer*s (IEEE) [4] u *Open Network Foundation (ONF)* [5] respaldan su desarrollo.

El mundo de las redes está realizando un cambio más rápido de lo que se podría esperar y el futuro de las redes SDN es muy alentador. La posibilidad de programar y automatizar una red abre un sinfín de posibilidades para la industria y el personal IT. La puesta en escena de otras tecnologías ajenas a este trabajo como Big Data, IoT o IA secundan la necesidad de redirigir las redes a un nuevo plano de control que tenga independencia del hardware de fabricante.

En conclusión, y en base a lo expuesto, se considera razón suficiente para justificar la realización de un proyecto propio sobre esta tecnología.

1.2 Objetivos del Trabajo

El principal objetivo para la realización de este trabajo es estudiar y analizar las redes overlay o superpuestas dentro del paradigma SDN. Conocer su tecnología, protocolos, arquitecturas y los beneficios que pueden aportar a la gestión del tráfico en centros de datos de escala masiva (en adelante MSDC).

- Objetivos Principales:

  - Estudiar y analizar las tecnologías de redes Overlay y SDN
  - Realizar una comparativa de las tecnologías SDN
  - Crear un laboratorio virtual para el análisis de estas tecnologías.
  - Crear un laboratorio virtual para el despliegue de un controlador SDN.
  - Obtener una serie de conclusiones sobre la viabilidad de esta tecnología.

- Objetivos Secundarios:

  - Estudiar la evolución de los CPD y sus arquitecturas
  - Conocer sus limitaciones y causas de colapso.
  - Estudiar la integración de la computación en la nube y la virtualización en los CPD.
  - Analizar la demanda de tráfico y aplicaciones.
  - Conocer los principales fabricantes y organizaciones implicadas.
  - Analizar los nuevos estándares e implementación que están surgiendo.
  - Obtener una serie de conclusiones.

1.3 Enfoque y método seguido

El primer enfoque de este trabajo será el estudio de las tecnologías que engloban las redes superpuestas u Overlays. En este documento, bajo una base teórica inicial se pretende ayudar al lector a comprender los conceptos desarrollados en los primeros apartados de la memoria para que de forma clara y sencilla comprenda el alcance y uso de estas nuevas tecnologías.

El segundo enfoque de este trabajo será conocer y explicar el funcionamiento de estas tecnologías, una vez adquiridos los conocimientos básicos, para realizar diversas simulaciones en las que se ponga a prueba las características de configuración de cada una de ellas.

El tercer enfoque de este trabajo será implementar y explicar el funcionamiento de un controlador SDN bajo un entorno tanto gráfico como de programación. La elección del controlador a desplegar se realizará en función del análisis previo de software descrito en el apartado correspondiente de la presente memoria. Una vez adquiridos los conocimientos básicos de la herramienta, se realizarán las diversas simulaciones que permitan entender al lector la gestión de la red bajo el mando de un controlador definido por software.

1.4 Planificación del Trabajo

La planificación del trabajo se ha dividido en dos partes. Por un lado, se ha realizado la creación de un cronograma para obtener una visión general de las tareas y sus tiempos. Y, por otro lado, se ha realizado un diagrama de Gantt para proporcionar una visión más detallada y precisa.

- **CRONOGRAMA:**

Cronograma del proyecto en forma de tabla.

Este cronograma está detallado por semanas y, por lo tanto, no reflejan las horas exactas a que se dedica cada tarea, en su lugar, se definen los días totales de trabajo para cada una de ellas. El objetivo de este cronograma es tener una visión general de la planificación del trabajo que permita presentar el tiempo empleado. Concretamente, se puede observar las tareas que se realizan cada semana y que partes de la memoria corresponden a cada una. También se puede observar las entregas inmediatas o PECs que forman parte de la planificación.

ACTIVIDAD	INICIO	FIN	DURACIÓN
Inicio del Trabajo de Fin de Grado	28/09/2022	15/01/2023	109 días
PEC 1	**28/09/2022**	**16/10/2022**	
Planteamiento del proyecto	29/09/2022	30/09/2022	2 días
Planificación	01/10/2022	03/10/2022	3 días
Cronograma	04/10/2022	05/10/2022	2 día
PEC 2	**28/09/2022**	**19/11/2022**	
Contexto y justificación del trabajo	06/10/2022	08/10/2022	3 días
Objetivos y enfoque	09/10/2022	11/10/2022	3 días
Documentación sobre arquitecturas de centro de datos	12/10/2022	16/10/2022	5 días
Documentación sobre Cloud y virtualización	17/10/2022	21/10/2022	5 días
Documentación sobre SDN	22/10/2022	26/10/2022	5 días
Análisis de tecnologías y protocolos	27/10/2022	31/10/2022	5 días
Análisis de controladores SDN	01/11/2022	05/11/2022	5 días
Documentación sobre NFV	06/11/2022	10/11/2022	5 días

Análisis sobre software a utilizar	11/11/2022	13/11/2022	3 días
PEC 3	**28/09/2022**	**24/12/2022**	
Creación de Laboratorio 1	14/11/2022	16/11/2022	3 días
Implementación de controlador SDN en ONOS	17/11/2022	27/11/2022	10 días
Resultados	28/11/2022	29/11/2022	2 días
Creación de Laboratorio 2	30/11/2022	04/12/2022	5 días
Implementación de tecnologías en OpenVswitch	05/12/2022	15/12/2022	10 días
Resultados	16/12/2022	17/12/2022	12 días
Valoración Productiva	18/12/2022	20/12/2022	3 días
Entrega Final	**28/09/2022**	**15/01/2023**	
Redacción de la memoria final	21/12/2022	31/12/2022	10 días
Revisión u corrección de la memoria final	01/01/2023	05/01/2023	5 días
Video de la presentación	06/01/2023	08/01/2023	3 días
Power Point	09/01/2023	11/01/2023	3 días
Informe de autoevaluación	12/01/2023	13/01/2023	2 días
Final de Proyecto	**15/01/2023**	**15/01/2023**	

Tabla I – Cronograma del proyecto

- **DIAGRAMA DE GANTT:**

El objetivo del diagrama de Gantt es exponer el tiempo de dedicación previsto para la realización de las tareas definidas en el proyecto.

A continuación, en la Ilustacion 2, se muestra el diagrama de Gantt de forma detallada donde están definidas las tareas a realizar con su periodo de tiempo y horas reales empleadas. Se han introducido varios hitos para las entregas inmediatas o PECs. Se incluye al final de la memoria, la versión completa del diagrama de Gantt inicial y final para su análisis detallado.

Para la realización de la planificación, se han definido en días naturales con un total de cuatro horas por día aproximadamente.

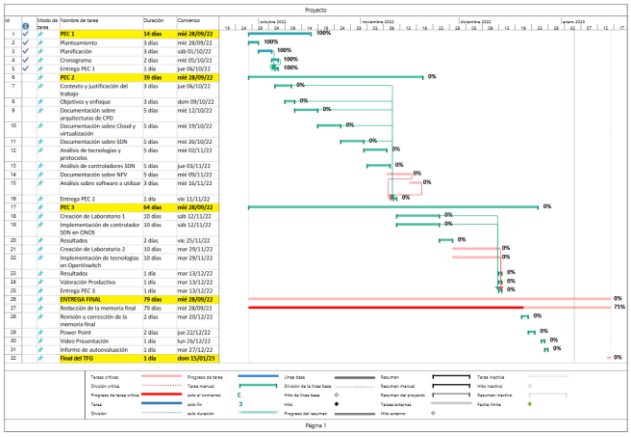


Ilustración 2 - Diagrama de Gantt inicial

1.5 Breve sumario de productos obtenidos

El presente trabajo consistirá en la realización de un estudio con base teórica y práctica de las distintas tecnologías existentes en las denominadas redes Overlay o superpuestas dentro del paradigma de las redes definidas por software. Es por este motivo, que el resultado o producto obtenido de este trabajo será la propia memoria.

No obstante, teniendo en cuenta el carácter de aplicación de este trabajo, se podría considerar de especial interés y como punto de partida, para cualquier empresa u organización que estuviera interesada en reducir sus gastos de capital o CAPEX, en inglés y sus gastos de operación u OPEX para generar nuevas oportunidades de negocio en la explotación de sus CPDs.

1.6 Breve descripción de los otros capítulos de la memoria

La memoria del proyecto se ha organizado en varios apartados, cada uno de ellos con una finalidad específica:

En el primer apartado que corresponde al punto **2.” Estado del Arte”** se hace referencia a la evolución de los CPDs y sus limitaciones actuales. La finalidad de este apartado es el análisis teórico de las tecnologías y arquitecturas en los CPDs bajo los estándares Tier.

En el segundo y tercer apartado que corresponde al punto **3.” SDN” y 4.” NFV”** se realiza un análisis teórico en profundidad sobre las tecnologías. La finalidad de estos dos apartados es la diferencia entre ambas, que, aunque parezcan tener puntos en común, presentan desigualdades notables que requieren de un estudio individualizado.

El quinto apartado **5. “CPDs MSDC”** tiene como propósito conocer las necesidades de los centros de datos de escala masiva y su simbiosis con SDN.

El sexto apartado y el núcleo de la memoria, corresponde al punto **6.” Overlay Networks”,** tiene como finalidad el estudio teórico de las distintas tecnologías que engloban a las redes superpuestas para su posterior análisis práctico.

El séptimo apartado, corresponde al punto **7.” Creación de un escenario virtual con controlador SDN”** tiene como finalidad, mediante un análisis previo de los apartados 3 y 4, el despliegue de un primer laboratorio virtual para un controlador SDN donde se analizará la creación y escalabilidad de las redes bajo su dominio.

En octavo apartado, corresponde al punto **8. “Creación de un escenario virtual para despliegue de tecnologías con OpenVSwitch”**, cuya funcionalidad será la creación de un laboratorio virtual para su posterior estudio.

Por último, el apartado **9.” Conclusión”** tiene como finalidad, la realización de una serie de deducciones con base en los apartados anteriores que fundamenten la presente memoria.

Posteriormente, se ha creado un apartado **9. “Valoración Productiva”** cuya finalidad es realizar un estudio económico del coste en despliegue de estas tecnologías en un centro de datos según el software previo a elegir.

El apartado,**10.” Glosario”** se presenta la definición de los términos y acrónimos más relevantes utilizados dentro de la presente memoria.

El apartado **11.” Bibliografía”** se aporta la lista de referencias bibliográficas consultadas para la realización de la presente memoria.

Y finalmente, el apartado **12.” Anexos”** se incluye toda aquella documentación que por su carácter extenso y no menos importante se considera esencial para el apoyo académico de la presente memoria.

2. Estado del Arte

Los CPDs albergan la potencia computacional, el almacenamiento y las aplicaciones necesarias para respaldar un negocio empresarial. La infraestructura del centro de procesado de datos es fundamental para la arquitectura TI y una buena planificación del diseño permite aumentar su productividad. En los apartados siguientes, se profundizará en el conocimiento de la infraestructura de un CPD, sus principales arquitecturas, sus limitaciones y sus necesidades actuales.

2.1 Evolución del centro de datos

Inicialmente los CPD los constituían los mainframes [7], computadoras de gran tamaño ubicadas en salas exclusivas con capacidades de cómputo. Los primeros CPD se remontan a la década de 1940, donde la primera computadora programable del mundo, denominada *Electronic Numerical Integrator and Computer (ENIAC)* [8] inicia la denominada tecnología computacional. Paralelamente *Bell Laboratories* [9] lanza en 1954 su primera computadora transistorizada llamada *TRADIC* [10] todos estos avances se realizaron bajo interés militar.

La tecnología computacional, se disparará a partir de los años 60 y 70 con el uso de la computación transistorizada y la aparición de los minicomputadores, que ayudará a los

CPDs a abandonar la esfera militar y permitir su transición al ámbito comercial con instalaciones en espacios más polivalente como edificios u oficinas de la mano de *International Business Machines* (IBM). En los años 80, la rápida innovación y proliferación de empresas informáticas como IBM, Intel Xerox, Apple o Sun Microsystems, iniciarán las bases de la informática personal y la computación paralela. [11] La tecnología de la información comenzará a convertirse en un gran contribuyente económico y crecerá a una escala sin precedentes. La interrupción de UNIX System establecerá el modelo de arquitectura "cliente-servidor".[12] Sin embargo, no será hasta principios de los años 90, con la llegada de Internet, cuando los usuarios comiencen a interactuar con servidores ubicados en CPDs de todo el mundo. La aparición de la computación distribuida [13], donde los datos y las aplicaciones se distribuyen entre sistemas dispares, conectados e integrados por servicios de red y estándares para funcionar como un solo entorno. Ha significado que el término CPD se use para referirse al departamento que tiene la responsabilidad de estos sistemas, independientemente de dónde estén ubicados. La computación Grid [14] y el concepto de Clúster, dio libertad al diseño de sistemas y aplicaciones a mediados del 2000.Este modelo de computación supuso una gran cuota de mercado y un considerable coste económico para las pequeñas y medianas empresas. El aumento del número de servidores en los CPDs aumentó su complejidad de gestión y mano de obra especializada. Corporaciones multinacionales como Amazon y Google pudieron ver esta necesidad de mercado y empezaron a desarrollar grandes centros que brindasen a las demás empresas una variedad de servicios y soluciones tecnológicas acorde a sus necesidades sin necesidad de invertir en infraestructura propia. La virtualización y el cloud computing que se analizará en el siguiente apartado, permitirá la utilización de recursos a través de internet, de forma flexible, y pagando únicamente por el consumo efectuado. La externalización del equipamiento empleado para soportar las operaciones, incluyendo los componentes de hardware de almacenamiento, servidores y red. Hace que el acceso a recursos como servidores, conexiones, almacenamiento o herramientas relacionadas con Internet, sea fácil y asequible, permitiendo a las empresas desarrollar un entorno de aplicaciones bajo demanda, en el que se paga solo por lo que se usa.

Hoy en día, la tendencia vuelve a ser minimizar la complejidad del CPD reduciendo el número de dispositivos a gestionar y optando por una infraestructura más automatizada, eficiente y programada mediante el uso del software.

2.2 Arquitecturas tradicionales de CPDs

Antes de profundizar en las diferentes arquitecturas de un CPD, analizaremos primero las normas existentes para su infraestructura. El estándar más ampliamente adoptado para el diseño y la infraestructura de un CPD es ANSI/TIA-942 [15], este estándar clasifica en cuatro categorías *Tier* los centros de datos según su porcentaje de redundancia, tolerancia a fallos y tiempo máximo que pueden estar inoperativos durante un año.

TIER	% Disponibilidad	% Parada	Tiempo anual de parada
TIER I	99,67%	0,33%	28,82 horas
TIER II	99,74%	0,25%	22,68 horas
TIER III	99, 982 %	0,02%	1,57 horas
TIER IV	100,00%	0,01%	52,56 minutos

Tabla II – Clasificación TIER ANSI/TIA-942

- Tier I: Infraestructura básica, ofrece protección limitada contra eventos físicos con una única ruta de distribución no redundante.

- Tier II: Infraestructura con capacidad redundante, ofrece una protección mejorada contra eventos físicos con una única ruta de distribución no redundante.

- Tier III: Infraestructura con mantenimiento simultáneo, ofrece protección contra prácticamente todos los eventos físicos, proporcionando componentes de capacidad redundante y múltiples rutas de distribución independientes. Cada componente se puede quitar o reemplazar sin interrumpir los servicios de los usuarios finales.

- Tier IV: Infraestructura tolerante a fallos, proporciona los niveles más altos de tolerancia a fallos y redundancia. Los componentes de capacidad redundante y múltiples rutas de distribución independientes permiten el mantenimiento simultáneo, y una interrupción en cualquier parte de la instalación no causa tiempo de inactividad.

La arquitectura de un CPD puede variar según el uso o tecnología a desplegar, sin embargo, existen tres capas de diseño que son la base para desplegar cualquier CPD actual.

- **Capa de acceso (N1)**: Proporciona acceso a la red y es donde se conectan los usuarios con sus equipos de trabajo.
- **Capa de distribución (N2):** Proporciona conectividad basada en políticas y control de límites entre las capas de acceso y central.
- **Capa central o Core (N3):** Proporciona transporte óptimo entre sitios y enrutamiento de alto rendimiento. Debido a la criticidad de la capa del núcleo, los principios de diseño del núcleo deben proporcionar un nivel adecuado de resiliencia que ofrezca la capacidad de recuperarse rápida y sin problemas después de cualquier evento de fallo en la red.

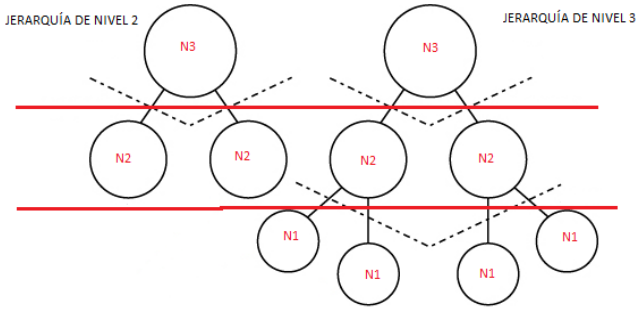
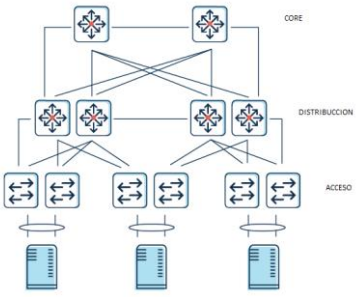


Ilustración 3 - Diseño jerárquico de nivel 2 y 3 elaboración propia

- **Arquitectura de red tradicional:**

Durante la última década, la arquitectura de red clásica de tres niveles ha sido la arquitectura de red predominante para los CPDs de mediana y gran escala, porque ofrecía el nivel deseado de flexibilidad de diseño, disponibilidad y rendimiento para las diversas aplicaciones en ese momento. La arquitectura general de una red de centro de datos o DCN en inglés, en un alto nivel, se basa en el mismo concepto de tres capas analizado anteriormente. Este modelo de arquitectura proporcionaba varias desventajas entre los que se incluye el uso del estándar IEEE 802.1D [16] definido como árbol de expansión o *Spanning-Tree* (STP). Este protocolo se caracteriza por desactivar automáticamente los enlaces de conexión, de forma que se garantice la eliminación de bucles en la red. Algunos de estos enlacen redundantes son necesarios si existen equipos como firewalls o balanceadores de carga que funcionen en modo activo-activo o activo-pasivo. Spanning-Tree, hace un uso terrible de la capacidad de la red e incluso con todas sus variantes es incapaz de resolver los problemas de fusión y convergencia, aumentando el TCO de un centro de datos.



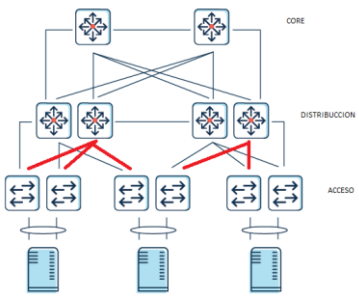


Ilustración 4 - Arquitectura tradicional de 3 niveles e Ilustración 5 - Bloqueo de enlaces con STP

- **Arquitectura de red 3 niveles con MC-LAG:**

La arquitectura de 3 niveles bajo el estándar IEEE 802.1ax-2008 [17] definido como Multichassis LAG o (MC-LAG) es una extensión de su antecesor, el estándar IEEE 802.3ad [18] definido como Link Aggregation Group o (LAG). Las implementaciones de MC-LAG son específicas del proveedor utilizado, por eso se conoce como VLT en Dell, vPC en Cisco o vLAG en Brocade. Las implementaciones propietarias deberán ser del mismo fabricante en capa de distribución y generalmente de las mismas características. En la capa de acceso se permite el uso multi-fabricante. La utilización de este protocolo permite el uso de varias interfaces desde la capa de distribución a la capa de acceso a través de una misma interfaz física o lógica.

Los grupos de agregación de enlaces multichasis (MC-LAG) permiten que un dispositivo forme una interfaz LAG lógica entre dos pares MC-LAG. Un MC-LAG proporciona redundancia y balanceo de carga entre dos pares de MC-LAG y una red de capa 2 sin bucles y sin necesidad de STP.

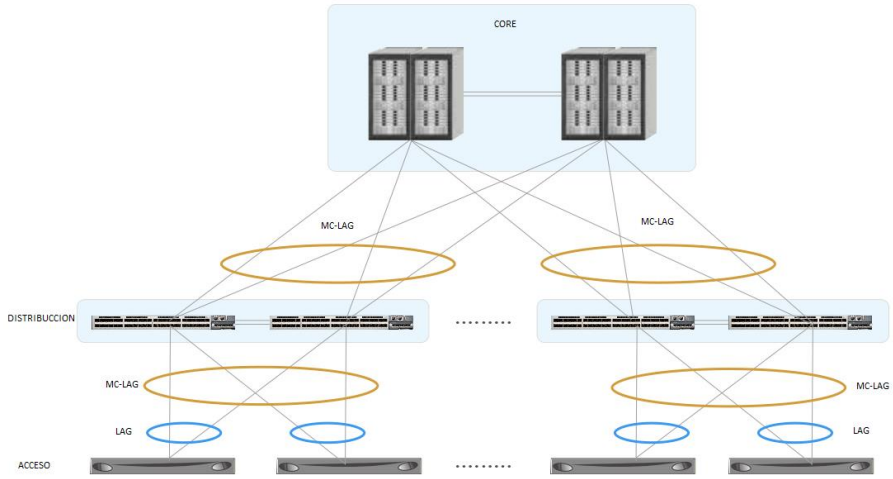


Ilustración 6 - Implementación de LAG y MC-LAG

En comparación con un diseño basado en STP, esta arquitectura ofrece una red más flexible y resistente con un rendimiento significativo y una optimización de la capacidad del ancho de banda. Sin embargo, la escalabilidad, es uno de los principales problemas de esta arquitectura para CPDs de gran escala con uso de cargas de trabajo virtualizadas. Por lo general, solo se pueden agrupar dos switches de agregación como un dispositivo lógico en los switches de acceso. Y, en consecuencia, los switches de acceso que se conectan a diferentes switches de agregación por enlaces ascendentes tienen que comunicarse a través del núcleo de capa 3, lo que puede romper los requisitos de movilidad de máquinas virtuales de capa 2 de extremo a extremo sin interrupciones o máquinas de carga de trabajo distribuidas en diferentes puntos de entrega o PODs cuyo término definiremos más adelante.
Como desventaja la arquitectura de nivel 3 con MC-LAG puede dejar a la red con un 50 % de su capacidad si ante una interrupción se pierde uno de switches de agregación que forman el anillo.

- **Arquitectura de red CLOS o Spine-Leaf:**

Las topología o arquitectura CLOS [19] llevan el nombre de su inventor, Charles Clos, un ingeniero de redes de telefonía que, en la década de 1950, estaba tratando de resolver un problema de crecimiento explosivo en redes telefónicas. En nuestra situación actual, con la expansión de los centros de procesamiento de datos y sus limitaciones de escalabilidad, nos enfrentamos a algo similar, cómo lidiar con el crecimiento explosivo de las redes DCN.

En la Ilustración 7, se muestra una topología Clos en su forma más simplificada. En el diagrama, los nodos de arriba son nodos *Spine* o columna y los de abajo son nodos de *Leaf* u hoja. Los nodos Spine se conectan a los nodos Leaf entre sí y viceversa [20]. En

los nodos Leaf se conectarán los servidores que con este modelo suelen estar a tres saltos de red de cualquier otro servidor cumpliendo las exigencias de cualquier aplicación de un CPD moderno. En este modelo de arquitectura, la gran variedad de enlaces existentes entre un servidor y otro permite hacer frente a las desconexiones o múltiples desconexiones sin provocar una pérdida total del servicio. En las arquitecturas anteriormente descritas con una redundancia bidireccional la pérdida de servicio era del 50%. En la arquitectura Clos la pérdida se reduce a una degradación en el ancho de banda de los enlaces. El ancho de banda entre los nodos es importante y se puede aumentar agregando tantos nodos Spine como sean necesarios con la única limitación del fabricante utilizado.

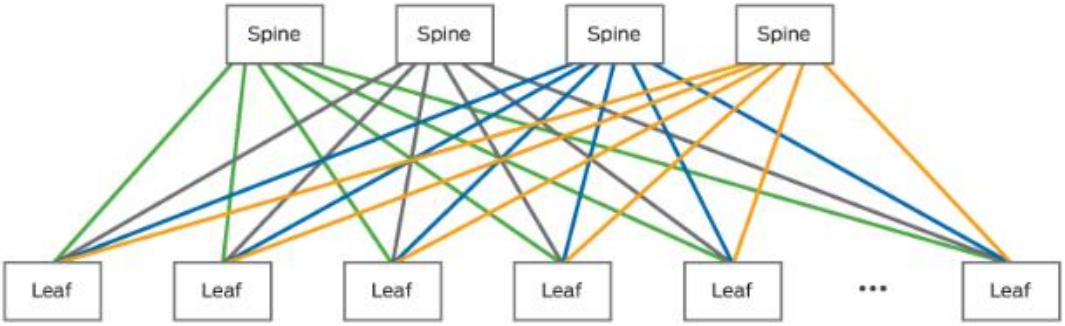


Ilustración 7 - Arquitectura CLOS o Spine-Leaf

Realizando un cálculo matemático para calcular la cantidad total de servidores que se pueden conectar a la red para una escalabilidad horizontal sin bloqueos. La fórmula matemática seria $n^2/2$ donde n es el número de puertos de un switch. Pongamos un ejemplo, para un switch de 64 puertos, la cantidad total de servidores que se podrían conectar sería de 64 * 64/2 = 2048 servidores. Si en su lugar tenemos un switch de 128 puertos, la cantidad total de servidores que se podría conectar sería 128 * 128/2 = 8192 servidores. En realidad, los servidores están interconectados a través de los nodos Leaf mediante enlaces de menor velocidad y los switches están interconectados por enlaces de mayor velocidad. Una implementación común es interconectar servidores a nodos Leaf a través de enlaces de 10 Gbps, mientras se interconectan switches entre sí a través de enlaces de 40 Gbps. Dado el aumento de los enlaces de 100 Gbps, una implementación prometedora es usar enlaces de 25 Gbps para interconectar servidores a nodos Leaf y enlaces de 100 Gbps para interconectar los switches. Pero debido a las restricciones de energía, la mayoría de las redes tienen como máximo 40 servidores y 32 puertos en un solo bastidor, aunque los nuevos diseños de servidores están superando este límite. Si realizamos el cálculo teniendo en cuenta las restricciones de energía actuales, la cantidad máxima de servidores que se puede conectar a una red leaf-spine simple es 40 * 32 = 1,280 servidores, una cantidad lo suficientemente grande para la mayoría de las pequeñas y medianas empresas.

Evolución de red CLOS o Spine-Leaf ≥ 3 niveles

La arquitectura Clos de tres niveles es una evolución de la arquitectura Clos simple. En esta arquitectura de tres niveles se pueden dar dos escenarios dependiendo si se agrega una capa superior de switches de Core por encima de los nodos Spine o una capa inferior de Pods o clústeres por debajo de los nodos Leaf. El término POD (punto de

entrega) representa un bloque estructurado estandarizado construido principalmente por recursos de red, computación y almacenamiento. En los CPDs, este concepto constituye la base para lograr un modularidad de diseño que pueda soportar requisitos heterogéneos de carga de trabajo de alto escalado horizontal y vertical.

Las empresas de servicios web no dudan en utilizar redes Clos de 4 o incluso 6 niveles para sortear las limitaciones. Al igual que la fórmula para un Clos simple, el número de servidores que pueden ser soportados por una red Clos de tres niveles construida son de puerto $n^3/4$. Para un switch de 64 puertos, el número total de servidores que se pueden admitir es $64^3/4 = 65.536$. Si recordamos, en un Clos simple, la cantidad de servidores que podrían ser compatibles era de 2.048. Este es un aumento significativo. Con switches de 128 puertos, podemos soportar $128^3/4 = 524.288$ servidores. Esta ecuación es válida para construir redes Clos de ≥ 3 niveles un número bastante elevado de servidores que solo las grandes corporaciones IT como Google, Meta u Amazon, por ejemplo, pueden desplegar.

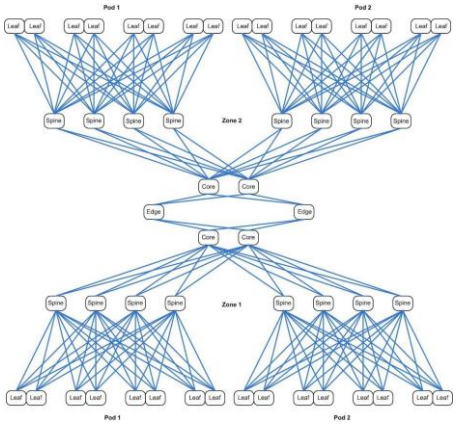


Ilustración 8 - Arquitectura CLOS o Spine-Leaf ≥ 3 niveles

Sin embargo, como en todas las arquitecturas hay limitaciones y en la arquitectura Clos surgen a raíz de las entradas ARP y las rutas multiples de enrutamiento (ECMP). Los switches Spine-Leaf soportan una cantidad muy alta de entradas ARP donde el número máximo de entradas varía según el fabricante utilizado. Por ejemplo, en equipos Brocade el número máximo de entradas ARP es de 256.000. Si bien esto puede parecer un gran número solo los switches más grandes pueden admitir tablas ARP de ese tamaño. En cuanto a la información de enrutamiento, la estructura Clos se basa en el uso de protocolos de transporte que ofrecen la capacidad de utilizar todos los enlaces disponibles entre cualquiera de los switches que forman la topología mediante el uso de rutas múltiples de igual costo (ECMP). El RFC 2992[21] analizó una estrategia particular de enrutamiento de rutas múltiples que implicaba la asignación de flujos mediante el hash de datos relacionados con el flujo en el encabezado del paquete. Esta solución fue diseñada para evitar estos problemas al enviar todos los paquetes de cualquier flujo de red en particular a través de la misma ruta mientras se equilibran múltiples flujos en múltiples rutas en general.
En la arquitectura Clos la solución a este problema es dividir la red en zonas. Donde cada zona tiene su propio conjunto de switches Spine y switches Leaf. Ampliando la figura 6, ahora podemos tener una red que se parece a la figura 7, donde se incluyen nodos de borde para mostrar cómo dos o más zonas se conectarían entre sí. El factor limitante aquí sería la disponibilidad de puertos en los nodos de borde.

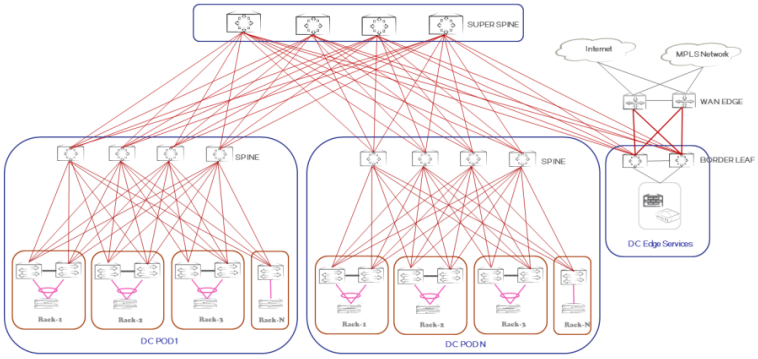


Ilustración 9 - Arquitectura CLOS con división por zonas

- **Arquitectura de red Fat-Tree:**

La arquitectura Fat-Tree es un caso especial de una red Clos que se puede denominar alternativamente como una red Clos plegada. La idea de topología fat-tree se propuso originalmente en 1985 por Charles E. Leirserson del Instituto Tecnológico de Massachusetts (MIT) [22] para aplicarse en el campo de la supercomputación. Sin embargo, a principios de 1997 la topología Fat-Tree fue ampliamente utilizada para redes de alto rendimiento en CPDs especializados [23].

El término Fat-Tree hace referencia a la cantidad de Pods que forman la topología. Estos Pods están números de izquierda a derecha mediante la numeración de Pod-0 a Pod (*k*-1). La topología consta de *k* pods con tres capas de switches:

- Switches de borde
- Switches de agregación
- Switches de núcleo.

Por lo tanto, en una topología fat-tree de tres capas (borde, distribución y núcleo) de *k* pod, hay *k* switches (cada uno con *k* puertos) donde:

- Cada Pod consta de $(k/2)^2$ servidores y 2 capas de *k/2* switches con *k* puertos
- Cada switch perimetral se conecta a *k/2* servidores y *k/2* switches de distribución.
- Cada switch de distribución se conecta a los switches *k/2* de borde y *k/2* de núcleo.
- Cada switch de núcleo $(k/2)^2$ se conecta a k pods

En la siguiente tabla se muestra la fórmula matemática para calcular el número de switches totales por cada capa, según el número inicial de Pods:

	Fat-tree with L levels	Two level Fat Tree L=2	Three Level Fat Tree L = 3	Four Level Fat Tree L=4
Number of Core switches	$\left(\frac{k}{2}\right)^{L-1}$	$\frac{k}{2}$	$\left(\frac{k}{2}\right)^{2}$	$\left(\frac{k}{2}\right)^{3}$
Number of Hosts supported	$2\left(\frac{k}{2}\right)^{L}$	$2\left(\frac{k}{2}\right)^{2}$	$2\left(\frac{k}{2}\right)^{3}$	$2\left(\frac{k}{2}\right)^{4}$
Total Switches	$(2L-1)\left(\frac{k}{2}\right)^{L-1}$	$3\left(\frac{k}{2}\right)$	$5\left(\frac{k}{2}\right)^{2}$	$7\left(\frac{k}{2}\right)^{2}$
Number of Edge Switches	$2\left(\frac{k}{2}\right)^{L-1}$	k	$2\left(\frac{k}{2}\right)^{2}$	$2\left(\frac{k}{2}\right)^{3}$
Number of Pods	$2\left(\frac{k}{2}\right)^{L-2}$	NA	k	$k^2/2$

Tabla III - Cálculo de Pods en arquitectura Fat-Tree

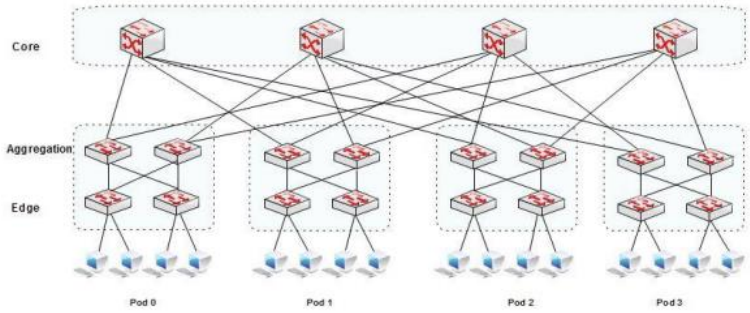


Ilustración 10 - Arquitectura Fat-Tree de 4 Pods

Las topologías Fat-Tree tienen ciertos beneficios. Todos los switches son semejantes con la misma cantidad de puertos, y cada puerto suele proporcionar la misma velocidad, para los equipos finales también se admite la misma velocidad. Cada host debe conectarse primero a un switch perimetral. En particular, en una topología fat-tree de 4 pods, hay dos rutas entre dos switches de borde dentro de un pod (intra-pod) y hay cuatro rutas entre dos switches de borde que se encuentran entre pods (inter-pod). Debido a que cada equipo final está conectado a un switch perimetral, la cantidad de rutas es la misma si dos equipos finales están asociados con dos switches perimetrales diferentes. Si dos equipos finales están fuera del mismo switch de borde, simplemente se crea una ruta entre estos dos equipos finales.

Tal y como se puede observar, en una arquitectura Fat-Tree no necesita recurrir al uso de un protocolo de enrutamiento, aun así, al igual que su homóloga la arquitectura Clos analizada anteriormente, requiere del uso de rutas múltiples de igual costo (ECMP).
La gran escalabilidad que permite esta arquitectura donde por cada switch de puerto k admite servidores $k^3/4$, es decir, si tenemos un switch con 64k puertos, podemos obtener un total de $64^3/4$ = 65.536 servidores es una cantidad considerable para aplicar en centros de datos de escala masiva, donde el número de puertos *k* de los switches suele ser superior a 64 interfaces.
Entre sus limitaciones están los cuellos de botella que se pueden producir entre la capa de borde y la capa de núcleo. Y la complejidad técnica del cableado.

Finalmente, se compara las características principales de cada uno de los modelos de arquitectura tradicionales de CPDs en la actualidad que se han analizado en esta sección.

	Arquitectura de 3 niveles con STP	Arquitectura de 3 niveles con MC-LAG	Arquitectura CLOS	Arquitectura Fat-Tree
Escalabilidad	Limitada. Por lo general, cada nodo de acceso está limitado a solo 2 nodos ascendentes.	Limitada. Por lo general, cada nodo de acceso está limitado a solo 2 nodos ascendentes.	Puede escalar hasta el límite del HW/SW de fabricante	Puede escalar hasta el límite del HW/SW de fabricante
Complejidad	Elevada gestión de STP	Moderada gestión de configuración y creacción de puertos LAG y MC-LAG	Simple gestión no requiere de STP o LAG	Moderada gestión de la configuración del tráfico entre capas
Comunicación Nivel 2	Con este modelo de diseño en grandes redes, los switches de acceso en diferentes POD pueden terminar comunicándose a través de la capa N3 Core	Con este modelo de diseño en grandes redes, los switches de acceso en diferentes POD pueden terminar comunicándose a través de la capa N3 Core	La naturaleza de la estructura de 2 niveles admite un gran escalamiento horizontal y en capa N2 de un extremo a otro entre los Leaf sin una ruta hacia la capa N3 Core.	Los switches se comunican entre diferentes POD y a través de la capa N3 Core.
Eficiencia en gestión de BW	Presenta limitaciones en la gestión del BW debido a la utilización de STP	Presenta limitaciones en la gestión del BW en enlaces upstream de agregación hacia la capa N3 Core	Eficiencia óptima. Se puede incrementar el BW aumentando la capacidad de los Leaf	Todos los switches de las diferentes capas y sus equipos finales tienen el mismo ancho de banda.
Disponibilidad	En caso de fallo del switch de agregación, generalmente habrá una pérdida del 50 % del BW disponible para cada uno de los switches de acceso afectados.	En caso de fallo del switch de agregación, generalmente habrá una pérdida del 50 % del BW disponible para cada uno de los switches de acceso afectados.	El fallo de un switch Spine reduce el BW disponible en el switch Leaf en solo un 25%	El fallo de capacidad de un switch solo afecta al 25 %
Convergencia	Muy lenta por el uso de STP	Rápida. El fallo de un miembro de enlace MC-LAG o switch tiene un impacto mínimo.	Rápida. El fallo de un switch Leaf o Spine tiene un impacto mínimo.	Rápida. Un fallo entre Pods tiene un impacto mínimo.
Comunicación de carga de trabajo	Carga de trabajo de escala limitada entre capa N2 y capa de N3	Carga de trabajo de escala limitada entre capa N2 y capa de N3	Carga de trabajo con soporte continuo de extremo a extremo	Carga de trabajo con soporte continuo de extremo a extremo
Idoneidad	Para CPDs de pequeña o mediana infraestructura.	Para CPDs de mediana infraestructura	Para CPDs de infraestructura masiva	Para CPDs de infraestructura de alto rendimiento.

Limitaciones	BW, conectividad, flexibilidad, cargas de trabajo en servidores.	BW, conectividad, flexibilidad, cargas de trabajo en servidores con o sin virtualización.	Cableados, comunicaciones entre routers de borde	Cuellos de botella entre capas. Cableado y gestión de rutas compleja.

Tabla IV - Comparación de Arquitecturas de (elaboración propia)

2.3 Necesidades actuales en CPDs de escala masiva.

Las redes de centros de datos modernos o DCN deben admitir una multitud de cargas de trabajo diversas y exigentes a bajo costo que incluso con las opciones de arquitectura más simples pueden afectar el rendimiento de las aplicaciones de misión crítica [24]. Esto obliga a los arquitectos de redes a evaluar continuamente las compensaciones entre los diseños ideales y las soluciones pragmáticas y rentables. En entornos comerciales reales, la cantidad de parámetros que el arquitecto puede controlar es bastante limitada y generalmente incluye solo la elección de topología, velocidades de enlace, sobresuscripción y tamaños de búfer de conmutación.

Los CPDs modernos a gran escala utilizan diseños de estructuras y clústeres, que son esencialmente topologías de Fat-Tree. Según Meta, el diseño Fabric tiene una mayor escalabilidad en comparación con el diseño de clúster.

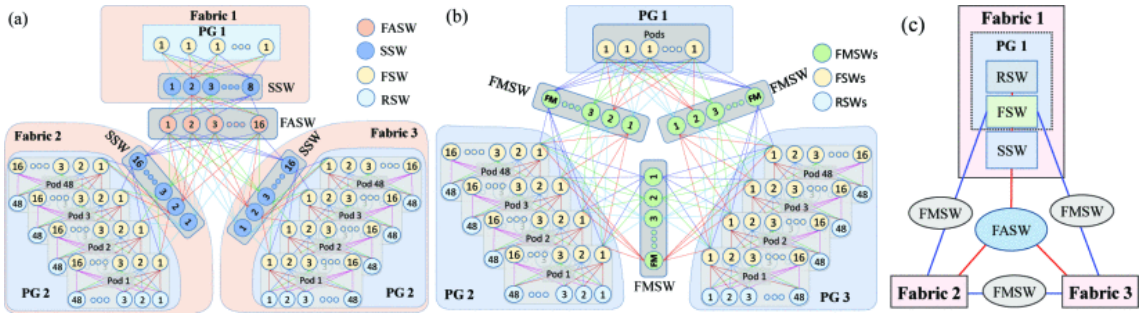


Ilustración 11 - Arquitectura F16 del Datacenter Meta

(a) Diseño interno de la topología F16 de Meta que conecta tres estructuras, (b) una topología de agregador de estructura de malla completa y (c) topología de agregador de estructura híbrida propuesta, que es la combinación de los diseños en las Figuras (a) y (b). Solo se muestran tres zonas/fabric para simplificar.

Las elecciones realizadas en el diseño de los CPDs modernos de escala masiva tienen consecuencias de gran alcance en la administración de la red. La más obvia es que dada la gran magnitud de la red, no es posible administrar manualmente los centros de datos de esta magnitud. Hay que tener en cuenta, que el cableado generado en una arquitectura Clos o Fat-Tree para conectar los switches es muy elevado y aumenta proporcionalmente. Si a esto le sumamos que el 30% de las incomunicaciones se producen por desgaste del cableado utilizado, la automatización es nada menos que un requisito para la supervivencia básica de cualquier CPD de escala masiva.

Para solucionar estos inconvenientes, las futuras arquitecturas deben ser capaces de crear patrones de diseño que permitan la automatización de los servicios. La demanda de las redes multi-inquilino, donde se necesita activar y desactivar conexiones virtuales rápidamente, no permite a las arquitecturas tradicionales que utilizan tecnologías basadas en VLAN o Spanning-Tree realizar estas operaciones rápidamente. Es un hecho, que una implementación tan rápida exige una automatización.

Del mismo modo, el despliegue de nuevos bastidores y los reemplazos de nodos en CPDs de escala masiva es superior a la de cualesquiera centros de datos con una arquitectura tradicional, por lo que su gestión manual, es prácticamente imposible y demanda el uso de tecnologías adecuadas capaces de cubrir estas necesidades de diseño.La superposición de las redes, la segregación y la utilización de software que permita la automatización y programación de los nodos de la red será el punto de partida para el despliegue de los nuevos centros de datos.

En los siguientes apartados se analizará el uso del software como solución para solventar las limitaciones actuales a nivel físico y como puede contribuir mediante las tecnologías virtuales a crear un CPD de gran escalabilidad.

2.4 Cloud computing y virtualización

El concepto *Network Cloud* o Nube se introdujo a principios de la década de 1990, donde a través de una capa de abstracción, se realizaba la entrega de datos a través de redes públicas. Sin embargo, no fue hasta el año 2006 donde el concepto computación en la nube, surgió en el ámbito comercial. Durante este año, Amazon lanzó sus servicios Elastic Compute Cloud (EC2) [25] que permitieron a las organizaciones arrendar capacidad informática y potencia de procesamiento para ejecutar sus aplicaciones empresariales. Google Apps también comenzó a proporcionar aplicaciones empresariales basadas en navegador ese mismo año y, tres años después, Google App Engine se convirtió en un hito histórico.

La definición que recibió la aceptación de toda la industria fue compuesta por el Instituto Nacional de Estándares y Tecnología (NIST). NIST publicó su definición original en 2009, seguida de una versión revisada que se publicó en septiembre de 2011, en este documento se proporcionaba una definición más concisa del concepto cloud computing:

"La computación en la nube es un modelo para habilitar el acceso de red ubicuo, conveniente y bajo demanda a una red compartida conjunto de recursos informáticos configurables (por ejemplo, redes, servidores, almacenamiento, aplicaciones y servicios) que se puede aprovisionar y liberar rápidamente con un mínimo esfuerzo de gestión o interacción con el proveedor de servicios". [26]

Las empresas necesitan la capacidad de adaptarse y evolucionar para afrontar con éxito los cambios provocados por factores tanto internos como externos. La agilidad organizacional es la capacidad de respuesta al cambio. Una empresa TI a menudo necesita responder a los cambios comerciales escalando sus recursos más allá del alcance de lo que se predijo o planificó previamente y la computación en la nube brinda esta capacidad de respuesta, de rentabilidad y evolución. Dado que cada parte de una red de telecomunicaciones moderna se puede diseñar para que sea totalmente IP,

también se puede transformar en una arquitectura *Network Cloud* o arquitectura de nube.

La arquitectura de nube hace referencia a la forma en la que se integran las distintas tecnologías para crear entornos TI que extraigan, agrupen y compartan los recursos de una red. La arquitectura de nube define la manera en la que se conectan todos los elementos. El estudio de arquitectura y servicios en cloud computing demanda un análisis más profundo, sin embargo, dada la naturaleza del proyecto, se dará especial relevancia al concepto de virtualización que analizaremos a continuación.

La primera virtualización convencional se realizó en mainframes de IBM en la década de 1960, pero Gerald J. Popek y Robert P. Goldberg codificaron el marco que describe los requisitos para que un sistema informático admita la virtualización. Su artículo de 1974 "*Formal requirements for virtualizable third generation architectures*"[27] describe las funciones y propiedades de las máquinas virtuales y los monitores de máquinas virtuales que todavía usamos en la actualidad.

Según su definición, una máquina virtual (VM) puede virtualizar todos los recursos de hardware, incluidos los procesadores, la memoria, el almacenamiento y la conectividad de red. Un monitor de máquina virtual (VMM) [28], que hoy en día se denomina comúnmente hipervisor, es el software que proporciona el entorno en el que operan las máquinas virtuales. La estructura de un hipervisor es bastante simple. Consiste en una capa de software que se encuentra entre el hardware y las máquinas virtuales. Hay dos clases de hipervisores, Tipo 1 y Tipo 2, donde el único elemento a destacar entre ellos es el modo en cómo se implementan.

- **Hipervisor Tipo 1:**

También denominado nativo, unhosted o bare metal. En este tipo de hipervisor el software se ejecuta directamente sobre el hardware. Además de tener mejores características de rendimiento, los hipervisores Tipo 1 también se consideran más seguros que los hipervisores Tipo 2. También requieren menos sobrecarga de procesamiento, lo que significa que se pueden ejecutar más máquinas virtuales en cada host. A nivel de seguridad [29], una máquina virtual solo puede dañarse a sí misma, provocando un solo bloqueo del invitado, pero ese evento no escapa a los límites del contenedor de la máquina virtual. Otros invitados continúan procesando y el hipervisor tampoco se ve afectado. Un invitado malintencionado, en el que el código intenta deliberadamente interferir con el hipervisor o con los demás invitados, no podría hacerlo.

- **Hipervisor Tipo 2:**

También denominado hosted. Este software se ejecuta sobre un sistema operativo anfitrión. A menudo, los hipervisores de tipo 2 son fáciles de instalar e implementar porque gran parte del trabajo de configuración del hardware, como las redes y el almacenamiento, ya ha sido cubierto por el sistema operativo. Los hipervisores de tipo 2 no son tan eficientes como los hipervisores de tipo 1 debido a esta capa adicional entre el propio hipervisor y el hardware. A nivel de seguridad, los hipervisores de tipo 2 también son menos confiables porque hay más puntos de falla: cualquier cosa que afecte la disponibilidad del sistema operativo subyacente también puede afectar al hipervisor y

a los invitados que admite. Por ejemplo, los parches estándar del sistema operativo que requieren un reinicio del sistema también forzarían el reinicio de todas las máquinas virtuales en ese host.

Ahora bien, como ya se comentaba anteriormente hay varios tipos de virtualización dependiendo de aquellos recursos se quieran virtualizar, en este proyecto se destacará la virtualización de funciones de red y contenedores.

- **Virtualización de las funciones de red**

La virtualización de las funciones de red o NFV, al que se le dedicará una sección en este trabajo, separa las funciones clave de una red (como los servicios de directorio, el uso compartido de archivos y la configuración de IP) para distribuirlas entre los entornos. Cuando las funciones del software se independizan de las máquinas físicas donde se alojaban, las funciones específicas pueden empaquetarse en una nueva red y asignarse a un entorno. La virtualización de redes, que se utiliza con frecuencia en el sector de las telecomunicaciones, reduce la cantidad de elementos físicos (como conmutadores, enrutadores, servidores, cables y centrales) que se necesitan para crear varias redes independientes.
Cada partición está lógicamente aislada de las demás y debe comportarse y aparecer como una red completamente dedicada para brindar privacidad, seguridad a un conjunto independiente de políticas, niveles de servicio o a decisiones de enrutamiento.
Con la virtualización de equipos de red podemos reemplazar a los routers de hardware por routers de software, y hacer lo mismo con cualquier otra pieza de hardware que pueda convertirse en software como switches, routers, firewalls, servidores, etc.

Actualmente, más del 80% de las aplicaciones se sirven en un entorno virtual y tanto los proveedores propietarios como de código abierto están implementando software virtual para sus dispositivos. Los grandes proveedores de cloud computing necesitan construir centros de datos masivos con miles de servidores y conmutadores de red. El argumento comercial para comprar y utilizar dispositivos de red de alto rendimiento no es viable económicamente y estas empresas comenzaron a innovar y construir sus propios mecanismos para la gestión de las redes. Un ejemplo de ello es Meta, que inició el *Open Compute Project* [30] y donó su diseño de hardware y software a la industria.

- **Virtualización de contenedores**

Un enfoque relativamente reciente de la virtualización se conoce como virtualización de contenedores.[31] En este enfoque, el software, conocido como contenedor de virtualización, se ejecuta sobre el núcleo del sistema operativo host y proporciona un entorno de ejecución para las aplicaciones. A diferencia de las máquinas virtuales basadas en hipervisor, los contenedores no pretenden emular servidores físicos. En cambio, todas las aplicaciones en contenedores en un host comparten un kernel de sistema operativo común. Esto elimina los recursos necesarios para ejecutar un sistema operativo independiente para cada aplicación y puede reducir en gran medida los gastos generales.

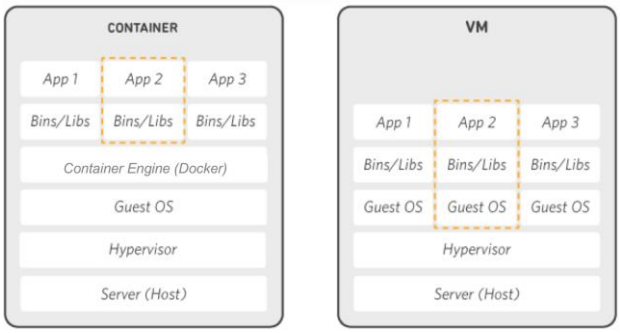


Ilustración 12 - Arquitectura diferencial entre contenedor y VM

3. NFV

3.1 ¿Qué es NFV?

La virtualización de funciones de red o (NFV) [32] es un área tecnológica de rápido crecimiento que está influyendo en gran medida en el mundo de las redes. Está cambiando la forma en que se diseñan, implementan y administran. Transformando la industria de las redes hacia un enfoque de virtualización, tal y como se analizaba en la sección anterior.

Para apreciar la motivación y la necesidad detrás de la rápida adopción de NFV por parte de la industria, es útil echar un vistazo a las demandas de los CPDs de escala masiva. Los CPDs a gran escala o MSDC en inglés, que alojan computación y almacenamiento, con su aumento factorial de dispositivos habilitados para datos y aplicaciones son solo algunos ejemplos de casuísticas que deben abordarse para mejorar el rendimiento y la latencia en las redes existentes. Esta sección, se analizará la forma en que NFV aporta una nueva perspectiva a las necesidades del mercado.

NFV se presentó por primera vez en el Congreso Mundial SDN OpenFlow en 2012[33] por un consorcio de proveedores de servicios que hicieron referencia a los principales desafíos a los que se enfrentan los operadores de red por su dependencia del hardware físico para desplegar su carta de servicios. El grupo propuso una forma de abordar estos desafíos y mejorar la eficiencia aprovechando la tecnología de virtualización. Para lograr este objetivo y definir un conjunto de especificaciones que harían posible pasar del enfoque tradicional a una red basada en NFV, siete operadores de telecomunicaciones como (AT&T,BT,Deutsche Telekom,Orange,Telecom Italia, Telefónica y Verizon) formaron un grupo de especificación ISG (Industry Specification Group) a través del Instituto Europeo de Normas de Telecomunicaciones (ETSI).[34] Hoy en día, está formada por más de 150 compañías, entre ellas las principales operadoras de telecomunicaciones, proveedores de infraestructura de telecomunicaciones y proveedores TI.
El acrónimo NFV, se usa como un término general para hacer referencia al ecosistema que comprenden los dispositivos de red virtual, las herramientas de administración y la infraestructura que integra estas piezas de software con el hardware de la computadora. Sin embargo, también podemos definir a NFV, como la tecnología que permite reemplazar dispositivos de red físicos que realizan funciones de red específicas con uno o más programas de software que ejecuten las mismas funciones de red. Un ejemplo, es reemplazar un dispositivo de firewall físico con una VM basada en software. Esta VM

proporciona las funciones de firewall, ejecuta el mismo sistema operativo y tiene la misma apariencia, pero con hardware genérico, compartido y no dedicado. Con NFV, las funciones de red se pueden implementar en cualquier hardware genérico que ofrezca los recursos básicos para el procesamiento, almacenamiento y transmisión de datos. La virtualización ha madurado hasta el punto de que puede enmascarar el dispositivo físico, lo que hace posible el uso de hardware comercial listo para usar, también denominado *Commercial Off-The-Shelf* (COTS). El hardware COTS se refiere a equipos informáticos, de almacenamiento y de redes de uso general que se construyen y se venden para cualquier uso, sin imponer un hardware o software privado.

Si echamos la vista atrás, históricamente el software de los dispositivos de red siempre ha sido propietario y personalizado, con CPUs dedicadas para procesar y reenviar el tráfico de red. Sin embargo, el hardware personalizado listo para usar (COTS) carece de estas CPU dedicadas de procesamiento y reenvío de paquetes. Para compensar la falta de CPU de reenvío y memoria caché de acceso rápido, se han desarrollado técnicas de software especiales para lograr un alto rendimiento al utilizar CPU de propósito general. Algunas de estas técnicas, como el Kit de desarrollo de paquetes distribuidos (DPDK) [35] de Intel y el Procesamiento de paquetes vectoriales (VPP) de Cisco son un ejemplo.

NFV proporciona cuatro beneficios principales a los proveedores de servicios de comunicaciones (CSP):

- **Flexibilidad**

La combinación de la rápida introducción de nuevos servicios y la nueva dinámica en el despliegue de servidores puede generar conjuntamente un aumento de los ingresos. La red se puede actualizar y adaptar a las necesidades reales de cada momento, tanto para ampliarla como para reducir su tamaño de forma sencilla e inmediata.

- **Reduce los gastos de capital (CAPEX)** [36]

La disminución de la utilización de hardware y la adopción de software de código abierto dan como resultado una reducción de los gastos de capital que se traducen en una reducción del consumo eléctrico y el consecuente ahorro energético. Como ejemplo, un estudio en conjunto entre las compañías Intel y British Telecom para comprender los beneficios de coste de un caso de uso de NFV mediante el despliegue de un equipo virtual en las instalaciones del cliente o vCPE. Se realizó mediante el uso de una herramienta de análisis integral o TCO (costo total de propiedad) para analizar las funciones de red física desde las instalaciones del cliente con traslado de servicios a la nube del operador. El resultado, fue una reducción del coste internos entre un 32% y un 39%. La cifra abarcó todos los costes, incluidos los costes de hardware, software, centro de datos, personal y comunicaciones. El análisis de TCO se realizó durante un período de cinco años e incluyó una variedad de funciones como firewall, enrutador, CGNAT (Carrier Grade NAT), SBC (Session Border Controller), VPN (Virtual Private Network) y WAN (Wide Area Optimización de la red).

- **Reduce los gastos operativos (OPEX)**

La automatización y la estandarización del hardware pueden reducir sustancialmente los gastos operativos. Ya que no es necesaria una contratación elevada de personal TI para realizar este tipo de tareas.

- **Agilidad**

La combinación de agilidad de servicio y autoservicio puede resultar en una mayor satisfacción del cliente. La optimización de la configuración y/o la topología de la red casi en tiempo real en función de los patrones de tráfico y demanda de servicios. Por ejemplo, la optimización de la ubicación y asignación de recursos a las funciones de la red de forma automática y casi en tiempo real podría brindar protección contra incomunicaciones de red sin diseñar una resiliencia completa.

Un punto para tener en cuenta es que, para brindar estos beneficios, NFV debe diseñarse e implementarse para cumplir con una serie de requisitos y desafíos técnicos, los cuales se incluyen en el Libro Blanco ISGN12: GSI. [37]

3.4 Aportaciones Open Source NFV

Para acelerar la transición a NFV, recientemente surgieron varios proyectos para orquestar VNF dentro de una red. Hay que tener en cuenta que para crear una pila NFV, es necesario integrar varios proyectos de código abierto. Sin embargo, los proyectos de código abierto no suelen realizar la integración y las pruebas, en conjunto, con otros proyectos. Además, aunque existen conjuntos de pruebas de proyectos individuales, ninguna comunidad u proveedor es responsable de las pruebas de rendimiento, funcionabilidad, estrés y longevidad de una pila integrada. Si a esto le sumamos que los proyectos individuales de código abierto suelen servir para múltiples casos de uso, no se centran en ningún caso exclusivo de uso para NFV.

A raíz de estos vacíos técnicos nació la comunidad Open Source NFV o también denominada OPNFV [44]. La creación de esta comunidad facilitó el desarrollo y la evolución de los componentes de NFV en varios ecosistemas de código abierto. A través de la integración, implementación y prueba a nivel de sistema, OPNFV creó una plataforma NFV de referencia para acelerar la transformación de las redes de empresas y proveedores de servicios.

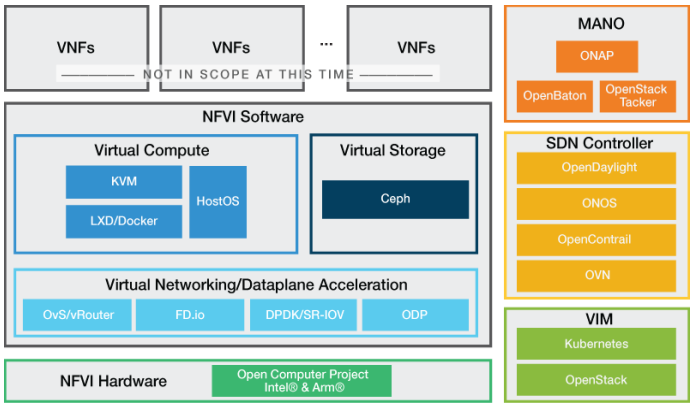


Ilustración 13- Proyectos upstream OPNFV

Otro proyecto opensource es ONAP [45] que se creó mediante la fusión de dos de las mayores iniciativas de redes de código abierto: ECOMP [46] y el proyecto Open Orchestrator (Open-O). Al aprovechar los beneficios de ambos proyectos, ONAP se basa en una arquitectura e implementación unificadas para ofrecer una plataforma abierta que permite a los usuarios finales crear sus propios VNF. La plataforma tiene como objetivo automatizar, orquestar y administrar VNF y servicios de red. ONAP incluye funciones de gestión de NFVO, VNFM, tiempo de diseño, garantía de servicio y FCAPS cualidades que corresponde a un orquestador MANO.

Todos estos proyectos integrados dentro de OPNFV y ONAP requerirían de un análisis en profundidad para su uso en NFV y SDN. Sin embargo, solo se profundizará en aquellos que nos sean de utilizan para alcanzar el objetivo final de este proyecto.

4. SDN

4.1 ¿Qué es SDN?

Si echamos la vista atrás, se puede observar que los dispositivos de hardware, más allá del aumento constante en las capacidades de la estructura de conmutación y las velocidades de interfaz o las comunicaciones de datos no han evolucionado mucho desde la llegada de las tecnologías IP, MPLS o móviles. IP y MPLS fueron tecnologías que permitieron a un operador de red crear redes y superposiciones de redes virtuales sobre esas redes base, de la misma manera que los operadores de centros de datos pudieron crear máquinas virtuales para ejecutarse sobre las máquinas físicas con la llegada de la virtualización. Con estas tecnologías de base emerge hace ya una década, un nuevo paradigma en el mundo de las redes denominada *Software Defined Network* o (SDN), que ofrecía una serie de características aptas para romper la barrera existente en los dispositivos de hardware.

Las redes definidas por software, no son una tecnología emergente actual, sino que surgieron del trabajo de investigación realizado inicialmente en 2004 como parte de la búsqueda de un nuevo paradigma de administración de redes, cuyo resultado fue el trabajo de la plataforma de control de enrutamiento (RCP 40) que se realizó entre las universidades de Princeton, Carnegie Mellon [47] y el trabajo de seguridad de red realizado al mismo tiempo como parte del proyecto SANE Ethene en la Universidad de Stanford y la Universidad de California en Berkeley. [48] Años más tarde, Google presenta en el *Open Summit* de 2012 un modelo de interconexión entre sus centros de datos como parte de la red WAN basada en SDN. [49]

Si analizamos el funcionamiento tradicional de las redes informáticas, podemos apreciar que utilizan tres planos separados para realizar sus tareas de conmutación; Estos planos se definen como, plano de datos, plano de control y plano de gestión.
El funcionamiento base del plano de datos es procesar los paquetes recibidos. Cuando llega un paquete, el plano de datos utiliza su información sobre el estado de reenvío local y la información contenida en el encabezado de cada paquete para tomar una decisión sobre si descartar el paquete o reenviarlo. Si el plano de datos decide reenviarlo, entonces debe decidir a qué computadora enviarlo y qué puerto en ese paquete de computadora debe recibirlo. Para seguir el ritmo de todos los paquetes que llegan, el procesamiento del plano de datos debe realizarse con extrema rapidez. Estos

tres planos se encuentran habitualmente acoplados a cada uno de los elementos que forman una red.

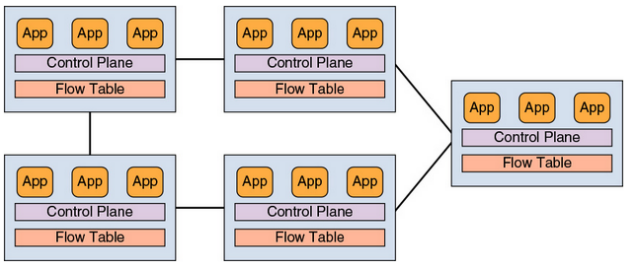


Ilustración 14 - Esquema reenvío red tradicional

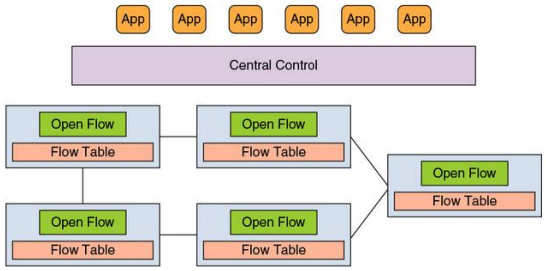


Ilustración 15 - Esquema reenvío red SDN

En la tecnología SDN, se realiza una separación del plano de control y el plano de datos para conseguir que el reenvío de paquetes no se realice en el mismo equipo. Esto se consigue aplicando software o programación de aplicaciones (APIs). Por lo que conceptualmente, se puede definir SDN como enfoque arquitectónico que permite la gestión de la red de manera inteligente y centralizada utilizando aplicaciones de software. Si bien, hay que dejar claro que SDN no es un conjunto de mecanismos, ya que SDN se puede implementar de varias maneras diferentes como se describen a continuación:

- **Rip-and-Replace Direct Fabric Programming**

En este método, los elementos de redes tradicionales deben reemplazarse con dispositivos habilitados para SDN puros. Un clúster de controladores SDN administrará los dispositivos de red y se integrará con otras plataformas. Algunos ejemplos son, Cisco ACI, BigCloud Fabric de BigSwitch Networks. En una red SDN tan pura, que está controlada por un controlador SDN, las tablas de reenvío de los dispositivos de red son administradas por el controlador SDN a través de un protocolo como OpenFlow que se estudiará en la sección 4.4. Los dispositivos de red no ejecutan ningún protocolo de enrutamiento ni toman ninguna decisión para reenviar un paquete o trama. En cambio, las tablas de hardware del conmutador están controladas por el controlador SDN.

- **Overlay**

Este método funciona con controladores SDN que construyen una red superpuesta SDN sobre una red heredada existente. La red subyacente desconoce la red superpuesta. Algunos ejemplos son, VMware NSX, OpenStack Neutron, Tungsten Fabric, etc. Los controladores SDN superpuestos se integran con plataformas de virtualización como VMware, Xen u OpenStack para construir una red superpuesta virtual para que las

cargas de trabajo virtuales se comuniquen entre sí y con el mundo exterior. Las redes superpuestas utilizan protocolos de encapsulación como VXLAN (Virtual Extensible Local Area Networks) para construir redes virtuales. En este trabajo se realizará un análisis exhaustivo de este tipo de implementación.

- **Híbrido**

Este modelo funciona con equipos de red que admiten OpenFlow como protocolo adicional dentro de su pila de software. Por ejemplo, un conmutador puede ejecutar BGP y OpenFlow juntos, lo que permite que un controlador SDN anule las tablas de reenvío de paquetes del dispositivo de red. Algunos ejemplos son, switches habilitados para OpenFlow de Brocade, HP. Este modelo, reúne el valor de las redes heredadas y SDN, ya que funciona con protocolos como OpenFlow y BGP-PCEP (BGP-Path Computation Element Communication Protocol).

El impacto de SDN y las redes modernas para los proveedores de servicios es más significativo, ya que puede ayudar a un proveedor de servicios a implementar una política de ingeniería de tráfico flexible en la red troncal central. Esto permite que la red central reenvíe el tráfico de la forma que desee el proveedor de servicios, en lugar de la forma que dicta el protocolo de enrutamiento. Hoy en día, las redes SDN cuentan con el apoyo de varias instituciones: Metro Ethernet Forum (MEF-SDN), Open Network Fundation, IEEE SDN, IETF I2RS. Otros organismos de estandarización como ATIS, Broadband Forum, ETSI, ITU-T, OIF u TM Forum están trabajando para extender a sus especificaciones los principios de uso de SDN.

En definitiva, las redes SDN tienen un gran potencial para cambiar la forma en que se integran las tecnologías IP, reduciendo la complejidad operativa, los costes y consiguiendo una red más eficiente combinada junto a NFV, tal y como explica ONF en su informe del año 2014.[50] Este informe demuestra como la simbiosis de SDN/NFV puede acelerar los despliegues de redes de forma escalable y elástica.

4.2 Diferencias entre SDN y NFV

Las principales diferencias entre SDN y NFV tienen que ver con las responsabilidades generales de administración de la red de enfoque y los estándares que guían el desarrollo arquitectónico y funcional.

- **Funcionalidades:**

El principal objetivo de SDN es separar el plano de datos y el plano de control sin necesidad de cambiar de software cada vez que se quiera hacer realizar un cambio. Mientras que en NFV, el principal objetivo, es la gestión de las funciones de red y su flexibilidad de adaptación, que permite realizar los cambios en cualquier momento según las necesidades de la red.

- **Ubicación:**

Otro de los puntos en los que difieren ambas tecnologías es la ubicación en la que se encuentran ya que SDN y NFV trabajan en ubicaciones distintas. El escenario de implementación en SDN son los centros de datos y el entorno de la nube. Mientas que la

implementación de NFV se realiza principalmente en las redes de los proveedores de servicios.

- **Capas Modelo OSI:**

A nivel de aplicación según el modelo OSI, SDN opera en la capa 2 (Datos) y la capa 3 (Red). NFV opera en la capa 4 (Transporte), capa 5(Sesión), capa 6 (Presentación) y capa 7 (Aplicación).

- **Estándares:**

Open Network Foundation busca desarrollar estándares abiertos, así como estándares independientes del proveedor, para la interfaz de comunicaciones definida entre las capas de control y reenvío de una arquitectura SDN.El Instituto Europeo de Estándares de Telecomunicaciones (ETSI) define y mantiene estándares aplicables a nivel mundial para las tecnologías de la información y las telecomunicaciones con respecto a NFV. En muchos sentidos, SDN y NFV son interdependientes, pero cuando se implementan juntos pueden lograr infraestructuras de red flexibles y ágiles.

En conclusión, NFV proporciona las funciones de red básicas y SDN asume la responsabilidad de gestión de alto nivel para orquestar las operaciones generales de la red.

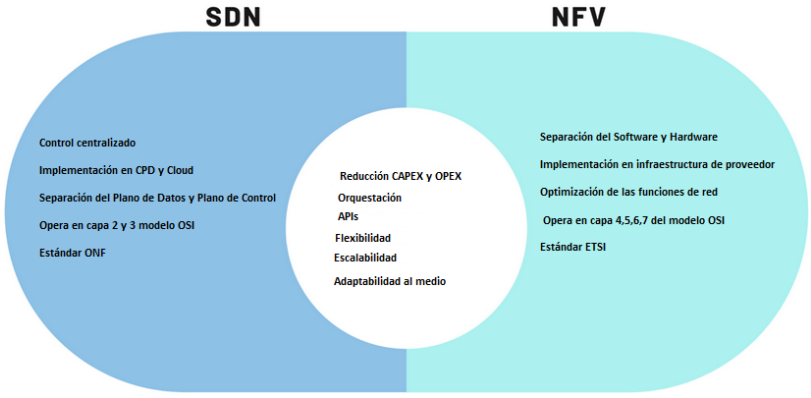


Ilustración 16 - Comparación SDN/NFV elaboración propia

4.3 Arquitectura SDN

La arquitectura de red SDN utiliza el desacoplamiento del plano de control y del plano de datos. Y utiliza en su lugar un controlador externo de software que con el uso de APIs programables permitían realizar los reenvíos a la red con independencia del hardware utilizado. La ONF define a la arquitectura SDN como dinámica, manejable, rentable y adaptable. Y separa la arquitectura SDN en tres niveles o capas:

Capa de Infraestructura (Nivel 1): Esta capa incluye los datos, switches y routers físicos que forman la red. El tráfico se mueve en función de las tablas de flujo que son gestionadas de forma centralizada.

Capa de Control (Nivel 2): Esta capa mediante la instalación de un controlador SDN centralizado basado en software, regula y gestiona el hardware de reenvío de la capa de infraestructura. Es lo que anteriormente solía realizar el plano de control en los switches y routers y que ahora en la arquitectura SDN se realiza de forma centralizada mediante el controlador SDN.

Capa de Aplicación (Nivel 3): Esta capa incluye las aplicaciones de red que en la arquitectura tradicional eran manejadas por los switches y routers. En la arquitectura SDN, las aplicaciones y los servicios aprovechan la capa de control e infraestructura. Conceptualmente, la capa de aplicación está por encima de la capa de control y permite un fácil desarrollo de la aplicación de red. Estas aplicaciones, al ser programables realizan todo tipo de tareas de administración. En un ejemplo de uso, un administrador puede crear un conjunto de reglas y aplicaciones de reenvío, a la vez que puede elegir que aplicaciones desea permitir o donde desea implementar esas funciones para un grupo de usuarios o conjunto independientemente del dispositivo a utilizar.

En la arquitectura SDN se puede virtualizar toda la red. El uso de software y el protocolo OpenFlow que analizaremos a continuación, permiten el control programable de cómo las máquinas virtuales se conectan entre sí. También permite crear agrupaciones lógicas e independientes de la ubicación de forma rápida y sencilla. Esto era algo complejo, lento y propenso a errores en las redes tradicionales donde se requería una configuración independiente por cada dispositivo.

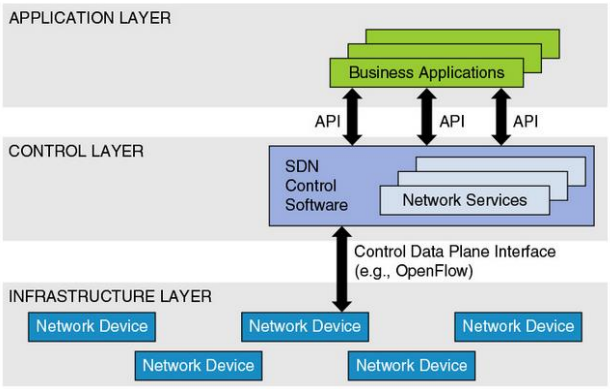


Ilustración 17 - Arquitectura SDN

4.4 Protocolo OpenFlow

El protocolo OpenFlow [51] es hoy en día el único protocolo de propósito general no patentado para programar el plano de reenvío en conmutadores SDN. Su área de función es gestionar las comunicaciones entre el plano de datos y el plano de control en la arquitectura SDN. La especificación OpenFlow ha estado evolucionando durante varios años. La primera versión de lanzamiento, la versión 1.0.0, apareció el 31 de diciembre de 2009, aunque antes de esa fecha existían numerosas versiones preliminares que se pusieron a disposición con fines experimentales a medida que evolucionaba la especificación. Para la ONF, OpenFlow sigue siendo el núcleo de su visión SDN para el futuro y es la entidad responsable de la evolución de la especificación OpenFlow. A partir del lanzamiento de la versión 1.1, la ONF publica y

gestiona directamente las revisiones del protocolo OpenFlow. En el momento de escribir este trabajo la versión actual del protocolo es la v1.5.1.

Al inicio de la sección 4.SDN, se analizaban las diferentes implementaciones de SDN en función de la naturaleza del controlador. Se analizaban las implementaciones en que los switches era puros (exclusivos para OpenFlow) donde no se heredaban funciones ni control integrado y dependía completamente del controlador para las decisiones de reenvío; e híbridos (habilitados para OpenFlow) donde se admitía la operación y el uso de los protocolos tradicionales.

El protocolo OpenFlow se expanden continuamente y cubre una variedad de campos como la virtualización de redes, la seguridad, el control de acceso y el equilibrio de carga. Sus escenarios de aplicación son amplios, pero en este proyecto solo se analizará el uso del protocolo OpenFlow en un entorno de red de centro de datos. Para analizar su funcionamiento se puede consultar el Anexo 12.1.

Entre las limitaciones del protocolo OpenFlow dado que su evolución en el ámbito de las redes definidas por software es rápida, es difícil profundizar si versiones posteriores podrán corregir las limitaciones actuales que presenta sobre todo en lo relacionado con los campos de coincidencia que están actualmente limitados por los encabezados de los paquetes. El DPI o Inspección profunda de paquetes en la que los campos de la carga útil del paquete se pueden usar para distinguir flujos, no se admite en OpenFlow estándar. Otra limitación es el posible retraso en el procesamiento si no hay una entrada coincidente en la tabla de flujo del switch OpenFlow, en cuyo caso el procesamiento del paquete exige que se envíe al controlador. Si bien este es solo un comportamiento predeterminado y el switch se puede programar fácilmente para manejar el paquete explícitamente, este retraso potencial es inherente al paradigma OpenFlow. También hay una serie de posibles vulnerabilidades de seguridad introducidas por OpenFlow, como la denegación de servicio distribuido (DDoS) que pueden hacer que el controlador no esté disponible o el fallo al implementar la autenticación del switch en el controlador.

Si bien la existencia de otros protocolos en el ecosistema de SDN como NETCONF, RESTCONF, OF-CONFIG, OVSDB son importantes para su aplicación independiente del valor del protocolo OpenFlow, he decido dar mayor peso a este último en este trabajo por ser el protocolo de propósito general apoyado por la ONF.

4.5 APIs SDN

Las API o (*Application Programming Interface)* [52] no deja de ser código de programación que crea un programador para que dos aplicaciones, por ejemplo, que normalmente se encuentran en dispositivos separados de la red puedan intercambiar datos.

En las redes definidas por software (SDN), las API de dirección norte o *Northbound* y las API de dirección sur o *Southbound* se utilizan para describir cómo funcionan las interfaces entre los diferentes planos de la arquitectura SDN.

Southboud APIs: Definen la forma en que el controlador SDN debe interactuar con el plano de datos para realizar ajustes en la red, de modo que pueda adaptarse mejor a los

cambios. OpenFlow tiene propia Southbound API que le permite agregar y eliminar entradas a la tabla de para que la red responda mejor a las demandas de tráfico en tiempo real. Otros ejemplos de APIs de interfaz sur son, NetConf, SNMP, CLI (Telnet / SSH) y P4.

Northbound APIs: Definen la forma en que el controlador SDN debe interactuar con el plano de aplicación. La idea es abstraer el funcionamiento interno de la red y permitir que los desarrolladores de aplicaciones puedan realizar cambios para adaptarse a las necesidades del momento. Las API de Northbound también se utilizan para integrar pilas de automatización, como Puppet, Chef, SaltStack, Ansible y CFEngine, así como plataformas de orquestación, como OpenStack, vCloudDirector de VMware o de Apache CloudStack.

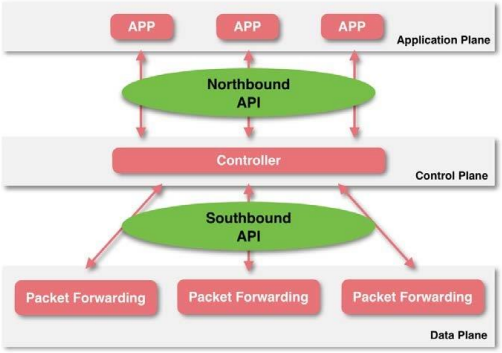


Ilustración 18 - Comunicación API entre Planos

4.6 Controladores SDN

En la arquitectura SDN el controlador es el elemento fundamental, la parte superior que actúa como el cerebro del sistema. En esta sección se analizarán las opciones comerciales y de código abierto actualmente disponibles.

4.6.1 Controlador SDN: OpenDaylight

El proyecto OpenDaylight (ODL) [54] se inició en 2013 cuando The Linux Foundation anunció la financiación de un marco de código abierto SDN liderado por la comunidad. Uno de los aspectos más interesantes del inicio de este proyecto fue que los miembros fundadores estaban formados por algunos de los proveedores más importantes, como IBM, Cisco, HP, Big Switch Networks, Arista Networks, Microsoft, Brocade y Juniper, entre otros y que generaron cierta controversia en la comunidad por sus intereses comerciales. Sin embargo, los ideales del proyecto ODL son todo lo contrario y promueven la adopción y la innovación de SDN a través de un marco común para crear código abierto que esté disponible al público.

La arquitectura del controlador ODL está basada en una API REST y utiliza YANG para generar los scripts de automatización. El controlador ODL actúa como una capa de abstracción de servicios que media entre las comunicaciones de las interfaces norte y sur. El administrador de topología almacena y administra la información sobre todos los dispositivos de la red. El administrador de estadísticas recopila y analiza datos para cada segmento de red. Las API hacia el norte también proporcionan datos sobre el puerto del switch, el medidor de flujo, la tabla y las estadísticas de grupo. Y el rastreador de host

almacena la información de todas las entidades de la red como su sistema operativo, tipo de dispositivo y versión.

Un dato curioso es como se implementa BGP sobre ODL. BGP se ejecuta y establece peering con otros vecinos intercambiando prefijos de red. La principal diferencia entre un BGP que se ejecuta en el controlador ODL y un BGP que se ejecuta en un router de red independiente es que en un router de red independiente o un switch, el BGP manipula y modifica directamente la tabla de enrutamiento del dispositivo. Sin embargo, en un controlador SDN, BGP almacena y mantiene una tabla BGP de forma centralizada, pero inyectará las entradas necesarias en la tabla de reenvío de varios switches subyacentes. Otra particularidad, es que ODL puede integrarse con OpenStack para crear redes aisladas para segmentos de red. Estos segmentos de red no son solo una representación de una VLAN, sino que están completamente aislados dentro de todo el dominio SDN.

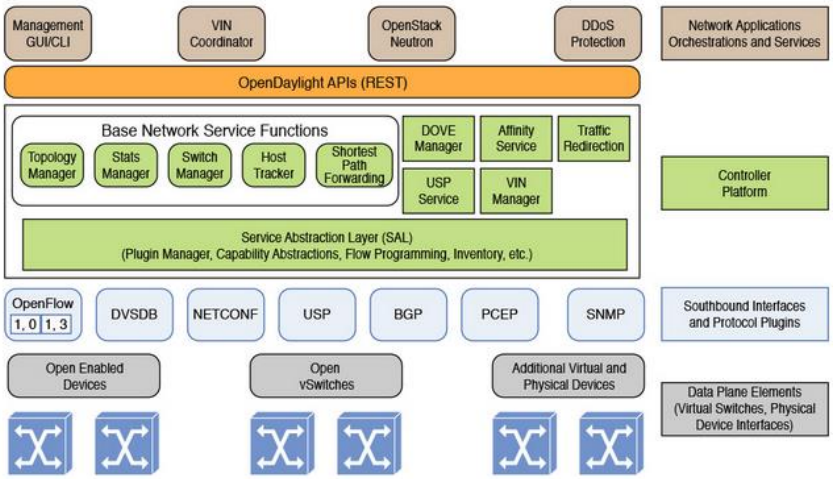


Ilustración 19 – Arquitectura ODL

4.6.2 Controlador SDN: ONOS

El proyecto ONOS u (*Open Network Operating System)* [55] está desarrollado por la ONF. Al igual que OpenDaylight, ONOS también está basado en una API REST y funciona sobre Apache Karaf OSGi. Su implementación puede llevarse a cabo en un clúster de varios servidores para brindar alta disponibilidad. ONOS admite varios protocolos Northbound para programar los switches y los routers tanto físicos como virtuales. ONOS admite protocolos como OpenFlow, NETCONF u OpenConfig, pero el sistema no está vinculado a ninguno de estos protocolos. Al igual que OpenDaylight, ONOS también proporciona un conjunto de abstracciones y modelos de alto nivel para las aplicaciones. Las aplicaciones pueden usar estos modelos para comunicarse con un dispositivo de red subyacente, como un switch, sin saber qué protocolo se usa para la comunicación con ese dispositivo. Por ejemplo, cuando una aplicación requiere una lista de direcciones MAC aprendidas de un switch, simplemente llama a una API en ONOS para obtener la lista. ONOS puede enviar un comando SNMP o NETCONF para obtener la lista de direcciones MAC del conmutador. Sin embargo, el método de comunicación es completamente transparente para las aplicaciones y los usuarios. No hay ninguna restricción para utilizar el modelo de abstracción. Las aplicaciones pueden

utilizar las bibliotecas de protocolos en dirección sur para comunicarse directamente con un dispositivo. Sin embargo, las aplicaciones estarán vinculadas para usar solo ese protocolo. Las aplicaciones se pueden cargar y descargar de forma dinámica, sin necesidad de reiniciar los servidores centrales de ONOS. La gestión de aplicaciones ONOS distribuye las aplicaciones de red de todos los demás nodos y servidores del clúster y garantiza que todos los servidores del clúster ejecuten esa aplicación. Además, NOS proporciona una GUI de interfaz web, una CLI y API REST para la gestión y la integración con otros sistemas.

La principal diferencia entre ONOS y ODL es que ONOS está más orientado a Telcos y proveedores de servicios, y ODL está más orientado a la nube y al centro de datos.

	OpenDayLight (ODL)	ONOS
Usos	ODL proporciona aplicaciones para integrarse con OpenStack o protocolos que se utilizan en centros de datos, como OVSDB	SONA (Arquitectura de red de superposición simplificada) es una aplicación ONOS que se integra con OpenStack y proporciona virtualización de red y aislamiento de redes
Aplicaciones	BGP-PCEP	CORD, Packet Optical, IP RAN, SDN IP, VPLS, Carrier Ethernet
Rendimiento y alta disponibilidad	ODL admite la agrupación en clústeres	La agrupación en clústeres de ONOS es madura y completa

Tabla V - Comparación entre ODL y ONOS (elaboración propia)

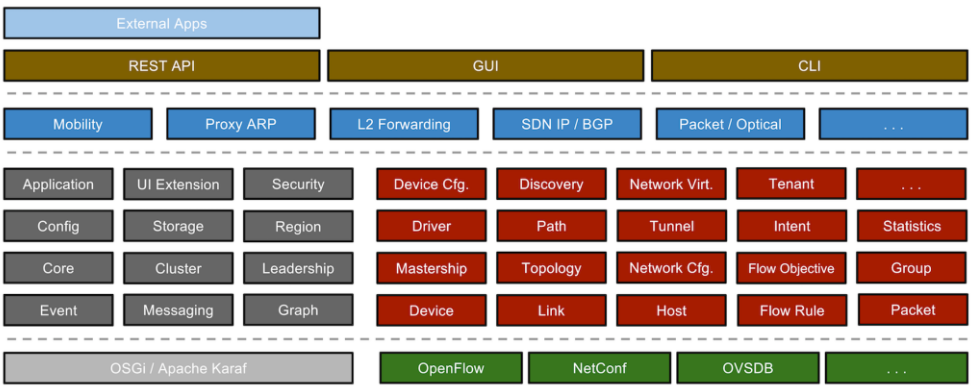


Ilustración 20- Arquitectura simplificada SDN ONOS

4.6.3 Controlador SDN: Tungsten Fabric

Tungsten Fabric [56]es un proyecto de código abierto con licencia de Apache 2.0 que fue desarrollado por Juniper Networks para proporcionar una solución completa de virtualización de red y SDN. Utiliza estándares de la industria de redes, como el plano de control BGP EVPN y las superposiciones de VXLAN para conectar sin problemas las cargas de trabajo en diferentes dominios de orquestadores. Por ejemplo, máquinas virtuales administradas por VMware vCenter y contenedores administrados por

Kubernetes. A medida que la virtualización se convierte en una tecnología clave para la prestación de servicios de nube pública y privada, los problemas de escala de la red se hacen evidentes con las tecnologías de virtualización que se han utilizado ampliamente hasta la fecha, por ejemplo, VMware con redes L2 y OpenStack con Nova, Neutron o redes ML2. Tungsten Fabric proporciona una plataforma de red virtual altamente escalable que está diseñada para admitir redes de múltiples inquilinos en los entornos más grandes al mismo tiempo que admite múltiples orquestadores simultáneamente.

Tungsten Fabric proporciona muchas características mejoradas sobre las implementaciones de redes nativas de los orquestadores y admite etiquetas MPLS o VXLAN VNI para identificar el tráfico de red superpuesto.

Todos estos controladores son ejemplos opensource, hay muchos otros como Ryu, NOX/POX, NodeFlow, Floodlight, pero he considerado especialmente estos tres controladores anteriores analizados, por la gran diversidad de compatibilidades con las plataformas Openstack, Vmware y Kubernetes.

Del mismo modo, en la industria de las redes SDN los fabricantes de equipos de telecomunicaciones como Cisco, HP, Nokia, VMware.etc. han comercializado sus propios controladores SDN adaptándolos a sus equipos bajo un software propietario. Algunos ejemplos de controladores SDN propietarios son:

- **HP VAN** [57]**:** Controlador SDN de redes de aplicaciones virtuales que utiliza OpenFlow para controlar las decisiones y políticas de reenvío en una topología SDN.

- **NEC ProgrammableFlow PF6800** [58]**:** El controlador NEC es programable y estandarizado, ya que se integra con OpenFlow y Microsoft System Virtual Machine Manager.

- **VMware NSX** [59]**:** Este producto de VMware es un sistema de control distribuido que puede controlar redes virtuales y superponer túneles de transporte sobre la infraestructura existente. El controlador se comunica con las aplicaciones para determinar sus requisitos antes de configurar todos los vSwitches.

- **Nuage Nokia** [60]**:** El controlador SDN Nuage virtualiza la infraestructura de red de cualquier data center y conecta automáticamente los recursos informáticos una vez implementada.

- **Cisco Viptela** [61]**:** Controlador enfocado a redes SD-WAN que proporciona capacidades avanzadas de enrutamiento, segmentación y seguridad para interconectar redes empresariales complejas con orquestación y políticas centralizadas.

Dado el carácter de código abierto en la que se orienta este proyecto y cumpliendo con los estándares de los principales organismos en SDN, se dará predilección a los controladores basados en código abierto por encima de los propietarios cuando se realice el posterior despliegue en un entorno de laboratorio.

5. SDN en CPDs MSDC

Anteriormente en la sección 2.3 del presente proyecto se analizaban las necesidades, tecnologías y nuevos entornos de los CPDs de escala masiva [62]. Los desbordamientos de las tablas de direcciones MAC, el agotamiento de VLANs, los entornos multi-inquilino, la ingeniería de tráfico, la eficiencia de las rutas, todo ello demandaba el uso de las redes definidas por software para paliar las limitaciones de las arquitecturas tradicionales y las redes heredadas. En esta sección se analizarán casos de uso específicos de SDN y SDN a través de superposiciones para abordar los requisitos del centro de datos actuales.

Con SDN y el protocolo OpenFlow, se inicia la introducción de la tecnología definida por software en centros de datos de gran escala. Los conmutadores OpenFlow pueden funcionar con la plataforma de gestión de la nube para implementar la asignación dinámica de recursos de red y la transmisión de tráfico de red bajo demanda, cumpliendo los requisitos de virtualización de red de los servicios en la nube y mejorando el rendimiento. Dado que los centros de datos tienen mucho tráfico, puede producirse una congestión si las rutas de transmisión no se asignan correctamente, lo que dificulta la eficiencia operativa. Para evitar esta situación, se puede implementar OpenFlow en los centros de datos para obtener de forma dinámica la información de transmisión de tráfico de cada enlace y entregar entradas de flujo para equilibrar la carga del tráfico entre enlaces. En cuanto al desbordamiento de las tablas MAC y el agotamiento VLAN, existe un pilar base en la nueva arquitectura del CPD SDN que aborda muchas de las necesidades mediante técnicas de tunelización y virtualización.

A continuación, se describirán casos de uso de SDN a través de tres posibles soluciones de implementación en CPD MSDC, mediante unas arquitecturas Open SDN, API SDN y SDN Overlay. Más adelante, en la sección 6 se definirán las tecnologías que sustentan estas arquitecturas.

5.1 SDN Overlay

El atributo principal de SDN Overlay [63] es la virtualización. Los dispositivos de red que gestionan suelen ser los switches virtuales que se ejecutan en hipervisores. En una topología de CPD las únicas direcciones MAC visibles a través de la red física son las direcciones MAC de los puntos finales del túnel, que se encuentran en los hipervisores. Como ejemplo, si hay ocho máquinas virtuales por hipervisor, habrá solo ocho direcciones MAC. Si los puntos finales del túnel están en nodos superiores o la cantidad de máquinas virtuales por hipervisor es mayor, el ahorro de direcciones MAC aún es mayor. En cuanto al problema de agotamiento de VLANs. SDN Overlay lo corrige debido a que usa tunelizados para las comunicaciones multiusuario en vez de utilizar VLANs.La cantidad de redes o segmentos tunelizados puede ser de 16 millones o más utilizando tecnologías de túneles como VXLAN, NVGRE o STT que analizaremos posteriormente.

La tecnología de superposición resuelve el problema de multiusuario por su propia naturaleza a través de la creación de redes virtuales que se ejecutan sobre la red física. Estos sustituyen a las VLAN como medio para proporcionar separación y aislamiento del tráfico. En las tecnologías de superposición, las VLAN solo son relevantes dentro de

un único inquilino. Para cada inquilino, todavía existe el límite de 4094 VLANs, suficiente actualmente para el tráfico de un solo inquilino, pero insuficiente para entornos MSDC.

Debido a que no se ocupa en absoluto de la red física debajo de ella, la tecnología de superposición ofrece poco en términos de mejorar los métodos de recuperación de incomunicaciones en el centro de datos. Si hay incomunicaciones en la infraestructura física, deben ser tratadas a través de los mecanismos ya establecidos, además de las superposiciones ya que las interacciones entre las topologías virtuales y físicas pueden ser difíciles de diagnosticar. SDN a través de superposiciones no tiene mucho que aportar en esta área debido al hecho de que no afecta a la infraestructura subyacente. Las cargas de tráfico a medida que pasan de un enlace a otro a través de la red no forman parte de la discusión sobre el tráfico que se canaliza a través de la parte superior de las interfaces y los dispositivos físicos. Por lo tanto, SDN a través de superposiciones depende de la tecnología de red subyacente existente para abordar este tipo de problemas.

Por lo tanto, a modo de resumen podemos afirmar que:

- SDN Overlay está diseñada para resolver problemas dentro del centro de datos.
- SDN Overlay crea una red de superposición virtual y, por lo tanto, permite superar las limitaciones de la red y mejorar la agilidad.
- SDN Overlay no soluciona en sí ningún problema relacionado con el comportamiento de la infraestructura física.

5.2 Open SDN

El atributo principal de Open SDN [64] es la utilización de un controlador centralizado. Este controlador centralizado es adecuado para crear una solución de una instancia de SDN a través de superposiciones. El controlador SDN puede crear túneles según sea necesario y convertirlos en puntos finales, o, dicho de otro modo, crea los túneles superpuestos según sea necesario y utilizar las reglas de OpenFlow para insertar paquetes en los túneles creados.
Dado que existe hardware que tiene soporte de túnel incorporado, se pueden construir dispositivos SDN que obtengan estos beneficios de la tunelización, pero con la ganancia de rendimiento del hardware. Por lo tanto, Open SDN puede resolver estas limitaciones de red de manera similar a la alternativa de SDN Overlay.

Además de las ventajas de las redes virtuales a través de túneles, Open SDN ofrece la posibilidad de cambiar la configuración y el funcionamiento de la red subyacente. Ya que Open SDN está diseñado explícitamente para poder controlar directamente el tráfico de red y el flujo de datos. Estas cualidades son particularmente adecuadas para abordar las necesidades de ingeniería de tráfico y eficiencia de ruta altamente demandadas en los CPDs MSDC.

5.3 API SDN

La característica principal de SDN a través de las API es establecer la configuración de los dispositivos de red de una manera más simple y eficiente. Dado que esto se hace desde un controlador centralizado al igual que Open SDN, una solución que utilice API

SDN mejoraría la automatización y generaría un mayor grado de agilidad en la red. No obstante, las APIs heredadas generalmente no se diseñaron para manejar la ingeniería de tráfico y aun con el uso de las APIs extendidas que pueden mejorar la ingeniería de tráfico en dispositivos heredados con planos de control local, solo es factible mediante el uso de aplicaciones SDN que se ejecutan en un controlador basado en API como OpenDaylight, Contrail, o APIC-DC. Estos son ejemplos de API SDN [65] que utilizan APIs propietarias para fomentar la agilidad requerida en los centros de datos. Estas APIs pueden utilizar el conocimiento sobre la red de manera similar a Open SDN.

En conclusión, si analizamos en conjunto cada caso, se puede afirmar que SDN Overlay, es una buena solución y su transición es más fácil a corto plazo, mientras que Open SDN ofrece un uso más amplio y completo. De los tres modelos, API SDN, puede no ser una solución óptima en el centro de datos de arquitectura heredada por los problemas que arrastra en ingeniería de tráfico.

Ahora bien, entre Open SDN y SDN Overlay, mi opinión es que los ejemplos de MSDC están evolucionando hacia un modelo Open SDN, un ejemplo de ello es Google, que está implementando Open SDN en sus centros de datos.[64] Mientras que SDN Overlay basada en hipervisor puede tener una mayor aportación en los centros de datos de arquitecturas tradicionales. Algunos ejemplos de este tipo de implementación SDN Overlay son Rackspace y Azure.

6. Overlay Network (Red Superpuesta)

Las redes superpuestas u (*Overlay Network)* [66] no son un concepto nuevo, pero se están volviendo cada vez más comunes en el ámbito de las redes. Se puede definir a una red superpuesta como una red lógica virtual construida sobre una red subyacente utilizando tecnologías de virtualización de red. Aunque una red superpuesta puede compartir dispositivos y líneas de una red subyacente, los servicios en las redes superpuestas están desacoplados de las tecnologías de interconexión y de las redes físicas en la capa subyacente.

Los dispositivos en redes superpuestas están interconectados a través de enlaces lógicos, constituyendo topologías superpuestas. Los túneles se establecen entre dispositivos superpuestos interconectados. Al enviar un paquete de datos, un dispositivo agrega al paquete de datos un nuevo encabezado IP más un encabezado de túnel para proteger el encabezado IP original. Luego, el paquete de datos se reenvía en función del nuevo encabezado IP. Cuando otro dispositivo recibe el paquete de datos, el dispositivo elimina el nuevo encabezado IP y el encabezado del túnel para obtener el paquete de datos original. En este proceso, los dispositivos de la red superpuesta desconocen la red subyacente. Las redes superpuestas admiten varios protocolos y estándares de red que se analizarán en la subsección 6 de este proyecto.

Con la introducción de la tecnología de redes definidas por software, las redes superpuestas con un controlador SDN desplegado tienen las siguientes ventajas:

- La transmisión del tráfico no depende de líneas específicas. Las redes superpuestas utilizan tecnologías de tunelización para seleccionar de manera flexible diferentes enlaces subyacentes y utilizan múltiples métodos para garantizar una transmisión de tráfico estable.

- Se pueden establecer diferentes topologías virtuales en redes superpuestas según se requiera sin necesidad de modificar la red subyacente.

- Los enfoques de cifrado se pueden utilizar para garantizar la seguridad del tráfico privado en Internet.

- Se admiten el corte y la segmentación de la red. Se pueden separar diferentes servicios para lograr una asignación óptima de los recursos de la red.

- Se admite el reenvío de rutas múltiples. En las redes superpuestas, el tráfico se puede transmitir desde el origen hasta el destino a través de múltiples rutas para implementar el equilibrio de carga y maximizar la utilización del ancho de banda de los enlaces.

Actualmente, una variante de red superpuesta que se ha vuelto popular en los últimos años es SD-WAN. La tecnología SD-WAN se enfoca en la conectividad de sucursales y oficinas donde los túneles IPSec se crean sobre una red física subyacente de capa 3, su despliegue se realiza como un cable de banda ancha, fibra hasta el hogar o circuitos de acceso a Internet dedicados para después ser canalizados a través de una red troncal IP o MPLS. Este enfoque de superposición facilita una red homogénea que se puede automatizar a través de SDN proporcionando una mejor escalabilidad, visibilidad de la capa de aplicación, confiabilidad y capacidad de administración.

Otro ejemplo de uso, son las redes superpuestas del centro de datos que suelen proporcionar una estructura totalmente mallada de túneles a través de una capa IP subyacente de nivel 3 que proporciona virtualización. Es decir, una red lógica homogénea definida en software y abstraída del hardware físico de la capa subyacente. Esta superposición de red virtual se puede implementar dentro de un solo centro de datos o en múltiples centros de datos para proporcionar escalabilidad y mayor flexibilidad a la red, incluyendo la automatización a través de redes definidas por software (SDN). Con una superposición lógica virtualizada, los servicios de red ahora se pueden implementar en cuestión de minutos.

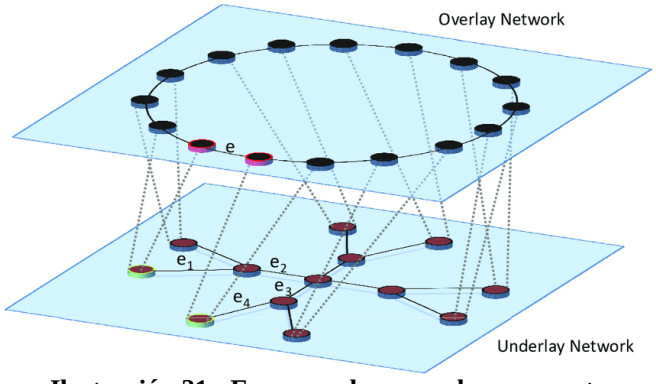


Ilustración 21 - Esquema de una red superpuesta

- **Arquitectura MAP-AND-ENCAP:**

Las superposiciones se basan en lo que comúnmente se conoce como arquitectura "map-and-encap".[67]. Se puede consultar el Anexo 12.2 para conocer su funcionamiento.

Otro aspecto clave de las superposiciones es el desacoplamiento de la vMAC (MAC virtual) y las direcciones IP utilizadas por las máquinas virtuales de la infraestructura de red física y las direcciones IP de infraestructura utilizadas por el centro de datos. Si una máquina virtual cambia de ubicación, los routers de borde superpuestos simplemente actualizan sus tablas de mapeo para reflejar la nueva ubicación de la máquina virtual dentro de la infraestructura del centro de datos. Esto es debido a que se usa una red superpuesta y la máquina virtual se puede ubicar en cualquier lugar del centro de datos en el que se encuentre la superposición. Las máquinas no tienen las limitaciones tradicionales impuestas por la red subyacente. Se puede apreciar las principales características que nos proporciona el uso de redes superpuestas en simbiosis con tecnologías SDN y la virtualización. En la siguiente tabla se muestra una comparación entre una red subyacente y una red superpuesta:

	Red subyacente	Red superpuesta
Transmisión de datos	Los datos se transmiten a través de dispositivos de red como enrutadores y conmutadores.	Los datos se transmiten a través de enlaces virtuales entre nodos.
Encapsulación de paquetes y sobrecarga	La encapsulación de paquetes se realiza en la Capa 2 y la Capa 3.	Los paquetes de datos deben encapsularse según el origen y el destino, lo que genera gastos generales adicionales.
Control de paquetes	Orientado al hardware	Orientado al software
Tiempo de implementación	La implementación de nuevos servicios implica una gran cantidad de configuraciones, lo que requiere mucho tiempo.	Cuando se implementan nuevos servicios, solo es necesario modificar la estructura de la topología de la red virtual, lo que permite una rápida implementación del servicio.
Reenvío de rutas múltiples	Debido a la baja escalabilidad, se requiere el reenvío de rutas múltiples, lo que aumenta la sobrecarga y la complejidad de la red.	Se admite el reenvío de rutas múltiples en redes virtuales.
Escalabilidad	La escalabilidad es pobre. Una vez que se construye la red subyacente, es difícil agregar nuevos dispositivos.	La escalabilidad es alta. Por ejemplo, una VLAN admite un máximo de 4096 discriminadores, mientras que una VXLAN proporciona un máximo de 16 millones de discriminadores.
Protocolos	Conmutación Ethernet, VLAN y protocolos de enrutamiento (OSPF, IS-IS y BGP)	VXLAN, NVGRE, SST, GRE, NVO3 y EVPN
Gestión de múltiples inquilinos	Se requiere aislamiento basado en NAT o VRF, lo cual es un gran desafío en redes a gran escala.	Se pueden administrar las direcciones IP superpuestas de varios inquilinos.

Tabla VI - Comparación entre red subyacente y superpuesta (elaboración propia)

6.1 Overlay Networks en MSDC

Las arquitecturas basadas en superposición proporcionan un nivel de no dirección que permite que los tamaños de las tablas de conmutación no aumenten en el orden de la cantidad de hosts finales admitidos por el centro de datos. Esto se aplica a los conmutadores tanto en el nivel de acceso como en el de agregación. En consecuencia, entre otras cosas, son ideales para abordar el requisito de gran escala exigido por los MSDC. En su forma más simple, se explicaba que una superposición no deja de ser un túnel dinámico entre dos puntos finales que permite transportar tramas entre esos puntos finales. Con la idea de base de la superposición y su arquitectura map-and-encap han surgido distintas clases de implementaciones superpuestas para su uso en los centros de datos.

- **Superposiciones basadas en red (Overlay networks):**

En las superposiciones basadas en la red, los switches de la capa de agregación no necesitan conocer el estado de los hosts finales a los que están conectados en el centro de datos, esto es debido a que solo escalan sus tablas a los switches de la capa de acceso a los que están conectados. En una topología compuesta por la capa de agregación y los switches de la capa de acceso, cada switch recibe un identificador único. Un switch dentro de un cableado en ToR [68] sirve como punto de conexión para todos los hosts finales debajo de él. Cada vez que un host final debajo de un ToR-1 necesita comunicarse con otro host final debajo de otro switch ToR-2. El ToR1 toma el paquete original y lo encapsula con un encabezado superpuesto donde los campos de origen y destino en el encabezado se establecen en identificadores ToR1 y ToR2, respectivamente. Este paquete encapsulado luego se envía a uno de los switches de agregación. Los switches en la capa de agregación solo necesitan dirigir el paquete hacia ToR2, lo que se puede hacer en función del encabezado de superposición. De esta forma, los switches de la capa de agregación desconocen completamente los hosts finales. Algunos ejemplos de superposiciones basadas en red son TRILL, FabricPath, SPB, OTV, LISP.

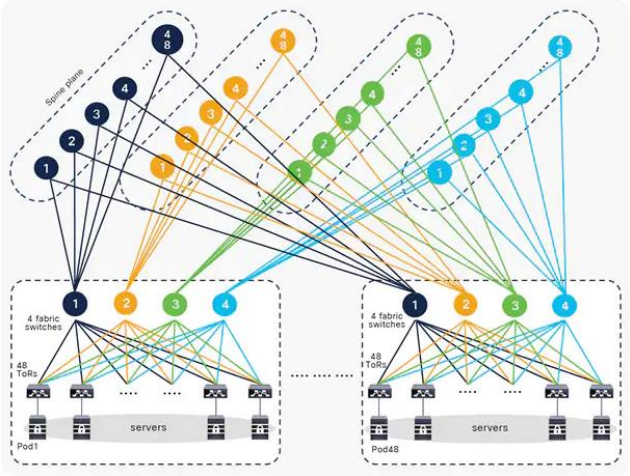


Ilustración 22 - Ejemplo MSDC con overlay basado en arquitectura fabric

Entre los beneficios del uso de superposiciones basadas en red se pueden destacar:

- Mejor utilización de los recursos de la red mediante el empleo de rutas múltiples para que los diferentes flujos puedan explotar las diferentes rutas redundantes disponibles entre los switches de origen y destino de la topología.

- Mejor resiliencia y convergencia más rápida debido a que si se cae un switch de agregación no se requiere de una gran cantidad de entradas de hosts finales en los switches de la capa de acceso. En su lugar, solo las tablas de conectividad superpuestas deben actualizarse para reflejar el cambio en la topología.

- Requieren muchas menos licencias, normalmente hay 20 o más servidores por switch y, por lo tanto, este enfoque suele ser mucho más rentable.

- Todas las superposiciones basadas en red ofrecen planos de control de redes definidas por software (SDN).

La única desventaja de las superposiciones basadas en red es la incapacidad de proporcionar segmentación para las máquinas virtuales que se ejecutan en el mismo host. Las superposiciones basadas en red pueden proporcionar una segmentación completa para todo el centro de datos, incluida la microsegmentación, pero si dos máquinas virtuales o contenedores están en el mismo host, para mover el tráfico a un segmento diferente o mantener el tráfico en el mismo segmento, es necesario que el tráfico de datos se envíe al switch leaf en la parte superior de la topología.

- **Superposiciones basadas en máquina (Overlay in VM):**

Las superposiciones basadas en máquina o host se originan en el hipervisor de una VM y nacen como resultado de la tecnología SDN en el centro de datos, pudiendo ofrecer una arquitectura de superposición completa de extremo a extremo. Esta opción es adecuada para arquitecturas completamente virtualizadas donde toda la topología de la red física se abstrae y se ve como una mera red de transporte generalmente basada en IP para la entrega de tramas encapsuladas. Una de las principales ventajas de las superposiciones basadas en máquina es que pueden realizar microsegmentación, incluso entre máquinas virtuales de un solo host, sin necesidad de que el tráfico de datos se envíe al switch leaf en la parte superior de la topología.

Las superposiciones basadas en host también vienen con una serie de advertencias en el sentido de que, si bien son atractivas desde el punto de vista de la agilidad y el aprovisionamiento rápido, la falta de conocimiento de la topología de red subyacente puede dar como resultado un reenvío de tráfico subóptimo que genera múltiples saltos de tráfico redundante a través de la red. Algunos ejemplos de superposiciones basadas en máquina son VXLAN, NVGRE y STT.

	Tipo de Tunelización	Fabricantes	Pros y Contras
Superposición basada en red o (Overlay Network)	Los túneles finalizan en el switch	Cisco, Juniper, Arista, Cumulus	Ventajas: Sin licencias por CPU o por máquina. VTEP acelerados por hardware. Puede agregar dispositivos no virtualizados en superposición. Desventajas: No hay segmentación para las VMs que residen en la misma máquina.

			El tráfico debe ir hacia el switch Leaf en la parte superior.
Superposición basada en máquina o host (Overlay VM)	Los túneles finalizan en el hipervisor de la VM	VMware NSX, Juniper Contrail, Nokia Nuage	Ventajas: Segmentación de red para VM en el mismo equipo. Desventajas: Licencias costosas. Complejidad de integración y operación significativa. Requiere de una gestión separada con aumento de CPU.

Tabla VII - Comparación tipos de superposiciones (elaboración propia)

Actualmente en estudio por la comunidad IEEE están las superposiciones híbridas [69], que son una combinación de superposiciones basadas en máquina y basadas en red que constan de una gran cantidad de encabezados superpuestos, para brindar mayor flexibilidad. La idea es que la superposición de máquina del switch virtual finalice en el switch de capa de acceso conectado directamente (ToR) y se inicie otra superposición que tenga accesibilidad en toda la topología. En el switch de la capa de acceso de salida conectado al destino, se realiza la operación inversa, es decir, se termina la superposición basada en la red y se marca la superposición basada en máquina (si corresponde), antes de enviar el paquete hacia el servidor de destino. Dicha arquitectura sirve para abordar algunas de las desventajas de las superposiciones puramente basadas en máquina y, al mismo tiempo, conserva sus características más destacadas. Un ejemplo de superposición híbrida que se analizará más adelante es la utilización de VXLAN con EVPN.

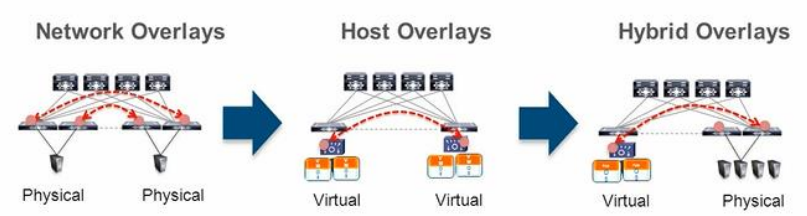


Ilustración 23 - Arquitecturas Overlays

6.2 Tecnologías

Los centros de datos modernos requieren de nuevos protocolos de red adaptados a las nuevas tecnologías de virtualización. Estos protocolos dependen de su propia naturaleza, de la arquitectura desplegada en el CPD y del carácter de implementación sobre los servicios a desplegar.

En la sección anterior, se explicaba la diferencia entre las superposiciones basadas en máquina y las superposiciones basadas en red, donde su uso estaba definido en el punto donde finalizaban sus túneles. La creación de estos tunelizados que se utilizan en los centros de datos, y el uso de nuevas tecnologías de tunelización (muchas anteriores a SDN) sirven para abordar las principales necesidades de los CPDs en el momento de escribir este proyecto.

Los protocolos de tunelización se basan en la idea de encapsular una trama MAC de capa 2 completa dentro de un paquete IP. Considerando esta razón de base, se puede dividir su uso de aplicación en dos tipos dependiendo de si su aplicación está realizada sobre superposiciones de red o superposiciones de máquina.

MAC-in-MAC [70]: También conocido como Provider Backbone Bridge (PBB), se define en el estándar IEEE 802.1ah. MAC-in-MAC es una técnica de red privada virtual (VPN) de capa 2. Encapsula la MAC del cliente en la MAC del proveedor de servicios, transmite la MAC interna como carga útil y, por lo tanto, mejora la capacidad de expansión de Ethernet y asegura los servicios. MAC-in_MAC se utiliza en supersposiciones de red y proporciona un túnel virtual para comunicar dos nodos Leaf. La capa subyacente sigue siendo una red de capa 2 al igual que la red superpuesta. La conectividad de capa 2 suele ser un requisito impuesto por los flujos de trabajo de las aplicaciones o los entornos de virtualización.
Para simplificar la arquitectura lógica del CPD, se puede utilizar cualquiera de los mecanismos citados en la Sección 2.2. MC-LAG se usa a menudo como un método de multiplexación inversa sobre múltiples enlaces que terminan en diferentes chasis físicos. Además de aumentar el ancho de banda y proporcionar redundancia de enlace, la variante de chasis múltiple agrega redundancia a nivel de nodo a la redundancia a nivel de enlace que proporciona LAG.TRILL y SPB son dos tecnologías diseñadas para evitar STP y proporcionar una red de capa 2 sin bucles. Ambos protocolos introducen un plano de control distribuido y usan IS-IS como protocolo de capa 2 para aprender la topología y calcular las mejores rutas en la red. Aunque su funcionamiento se describirá más adelante, ambos protocolos funcionan de diferentes maneras. El protocolo TRILL se comporta como un protocolo de capa 3, y el protocolo SPB que puede considerarse una optimización de PBB para mejorar la escalabilidad del núcleo basado en Ethernet se utiliza en capa 2.

MAC-in-IP [71]: Se utiliza en superposiciones de máquina, conceptualmente, el tráfico de la red virtual se ejecuta por encima de la infraestructura de la red física. Los hipervisores inyectan tráfico en la red virtual y reciben tráfico de ella. El tráfico de las redes virtuales pasa a través de esos dispositivos físicos, pero los puntos finales desconocen los detalles de la topología física, cómo se produce el enrutamiento u otras funciones básicas de la red. Dado que estas redes virtuales existen por encima de la infraestructura física, pueden ser controladas completamente por los dispositivos en el extremo de la red. En los centros de datos, estos suelen ser los hipervisores de las máquinas virtuales que se ejecutan en cada servidor. La arquitectura de la red subyacente utiliza protocolos de enrutamiento de capa 3 para la interconexión entre los nodos Leaf y Spine en el CPD. El uso de protocolos de enrutamiento de capa 3 elimina la necesidad de usar cualquier otro protocolo para evitar condiciones de bucle en la topología desplegada. Esta topología se puede utilizar como una red subyacente en caso de que fuera necesario extender el dominio de capa 2 entre servidores conectados a diferentes Leafs. VXLAN es un protocolo que se puede utilizar para configurar túneles superpuestos entre Leafs, lo que permite compartir la carga del enlace. Diferentes proveedores han establecido sus propios métodos para la tunelización de MAC-in-IP. Específicamente, Cisco ofrece VXLAN, Microsoft usa NVGRE y Nicira utiliza STT.

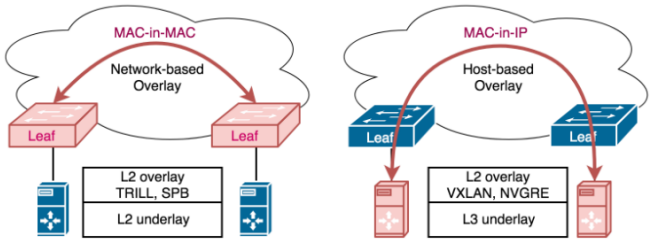


Ilustración 24 – Comparativa entre de Mac-in-Mac y Mac-in-IP

6.2.1 TRILL y SPB

TRILL – FabricPath:

El IETF ha estandarizado el protocolo TRILL abreviación de TRansparent Interconnection of Lots of Links bajo el estándar RFC6325[72] y permite manejar los bucles de una red de manera diferente a como lo hace Spanning Tree. TRILL maximiza el uso de todos los enlaces y permite usar características propias de los algoritmos de routing como el balanceo de carga. No obstante, aunque existe relación entre los dos protocolos, es importante señalar que hay una gran diferencia entre ellos y es que SPB no permite de ninguna manera la creación de bucles dentro de la red, pero TRILL sí.
TRILL combina las ventajas del comportamiento de reenvío de Capa 2 y Capa 3. Los switches que implementan TRILL son compatibles con los dispositivos IEEE 802.3 existentes. A modo de plano de control, TRILL utiliza IS-IS modificado para calcular la información de accesibilidad a los demás switches bajo su dominio. El uso de IS-IS, que es un protocolo de enrutamiento de estado de enlace permite el uso de ECMP y logra una convergencia más rápida durante la recuperación de incomunicaciones.
TRILL combina la simplicidad y la flexibilidad de la conmutación de capa 2 con la estabilidad, la escalabilidad y la capacidad de convergencia rápida del enrutamiento de capa 3, ya que no necesita configuración de direcciones IP. En su lugar, TRILL utiliza un campo TTL cuya misión es detectar y mitigar bucles intermitentes en la red. Todas estas ventajas hacen que TRILL sea muy adecuado para grandes redes de Capa 2 en centros de datos.

Los objetivos de TRILL son los siguientes:

- Prevención y detección de bucles
- Utilización eficiente de los enlaces.
- ECMP
- Rápida convergencia de red
- Configuración plug-and-play
- Protocolo de control unificado para unidifusión y multidifusión
- Evita el incremento de tablas MAC en la zona Core.
- Coexistencia con redes existentes de Capa 2 o Capa 3

Las tramas del protocolo TRILL incluyen TRILL Hello, LSP, CSNP, PSNP, MTU-probe y MTU-ack. Estas tramas de protocolo utilizan encapsulación 802.1Q y tienen una dirección de multicast de destino fija 0180-C200-0041. Los frames de datos TRILL tienen un formato específico, como se muestra en la Ilustración 31. Se agrega un encabezado TRILL y un encabezado Ethernet externo a la trama Ethernet original.

TRILL Ethernet frame - 1568 bytes/octets														
Pre SFD	O-DMAC	O-SMAC	O-TAG	Type	TRILL HEADER	I-SMAC	I-SMAC	I-TAG	C-TAG 0x8100	Type 0x0800	IP 20 bytes	TCP 20 bytes	Payload / Data 6-1460	CRC FCS	IFG
8	6	6	4	2	8	6	6	4	4	2	46-1500			4	12

Ilustración 25- Frame Trill

Dentro de una topología TRILL, los RBridge (RB) o bridge de enrutamiento son cualquier dispositivo que tenga TRILL implementado. Los RB se interconectan entre sí para formar una red TRILL. Dependiendo de la dirección del envío de paquetes, el RB que agrega un encabezado superpuesto o encapsula la trama es el RB de entrada o (IRB), y el RB que elimina el encabezado superpuesto o desencapsula la trama antes de enviarlo a una red que no sea TRILL es el RB de salida o (ERB). TRILL establece y mantiene las adyacencias entre los RB mediante la publicidad periódica de datagramas Hello, distribuye los LSP entre los RB vecinos y genera una LSDB para todos los RB de la red. Con base en la LSDB, cada RB utiliza el algoritmo SPF para calcular las entradas de reenvío destinados a otros RB.

SPB:

El estándar IEEE 802.1aq [73]define al protocolo Shortest Path Bridging (SPB) como Bridge de Ruta más Corta, este protocolo de capa reemplazó a los antiguos protocolos basados en STP (IEEE 802.1D STP, IEEE 802.1w RSTP, IEEE 802.1s MSTP) cuyo principal inconveniente era el mal uso de los enlaces ya que bloqueaban todos los enlaces alternativos a excepción del que estaba utilizando.

SPB al igual que TRILL proporción múltiples rutas de igual coste y utiliza IS-IS como protocolo de plano de control subyacente permitiendo que todos los enlaces en la topología estén activos. En SPB, los bridges utilizan IS-IS para encontrar la ruta más corta entre sí y calcular la topología. Pero a diferencia del enrutamiento, los bridges a gran escala utilizan solo el protocolo de estado de enlace IS-IS para la información topológica, no para la información de accesibilidad. Esto significa que las direcciones MAC no se anuncian dentro de IS-IS. (las direcciones MAC están ocultas para el Core de la red en una arquitectura CLOS). La escalabilidad de IS-IS para el anuncio de direcciones MAC es cuestionable para la implementación a gran escala; por lo tanto, tanto BGP para la distribución de direcciones MAC como IS-IS para la creación de topologías físicas pueden ser una buena opción de diseño para MSDC.

Shortest Path Bridging - VID (SPBV) y Shortest Path Bridging - MAC (SPBM) son dos variantes de SPB. Ambas variantes utilizan IS-IS como protocolo de topología de estado de enlace y ambas calculan rutas de camino más corto entre nodos. SPBV utiliza un ID de VLAN de camino más corto (SPVID) para designar la alcanzabilidad de los nodos y SPBM utiliza una combinación de MAC troncal (BMAC) e ID de VLAN troncal (BVID) para designar la alcanzabilidad nodal. Tanto SPBV como SPBM proporcionan interoperabilidad con Spanning Tree.

La principal diferencia entre TRILL y SPB está en su arquitectura de trama ya que SPB se basa en la trama Mac-in-Mac IEEE 802.1ah. En cuanto a TRILL, está claro que no es solo una tecnología de superposición que proporciona encapsulación y

desencapsulación ya que TRILL tiene su propio modelo de reenvío que incluye a los RBridges. TRILL aporta también un sistema plug-and-play y asignación automática de identificadores RB.

Si analizamos ambas tecnologías, siempre vamos a tener problemas relacionados con el software y el hardware debido a los cuales cualquier RBridge puede comportarse mal. Proporcionar una función plug-and-play es bueno para el administrador de red porque alivia la carga de la configuración manual. Pero al mismo tiempo, también debe ser adecuado para proporcionar suficiente visibilidad a la hora de solucionar problemas y monitorear la red.

	SPB	TRILL
Estándares	IEEE 802.1aq	IETF RFC6325
Prevención de Bucles	RPFC	TTL
IS-IS	IS-IS con extensiones TLV	Instancias de IS-IS con PDU
Elección de ruta	Con pre-provisión	Mediante el uso de RB
Encapsulación	Mac-in-Mac	Cabecera específica TRILL
Configuración de equipos	Manual	Plug-and-Play
Agregación de Puertos	Sí	No

Tabla VIII – Comparativa entre TRILL y SPB (elaboración propia)

6.2.2 VXLAN y NVGRE

VXLAN:

VXLAN o lan virtual extensible, es un protocolo estándarizado del IETF definido en el RFC7348[74]. VXLAN emplea un esquema de superposición MAC sobre IP/UDP incrementando el número de redes de capa 2 admitidas de 4096 a 16 millones. VXLAN fue diseñada principalmente para abordar los crecientes requisitos de escalabilidad de los MSDC permitiendo que los segmentos de Capa 2 se extiendan por todo el CPD e incluso entre CPDs. Esto es posible gracias a la superposición de redes virtuales de Capa 2 sobre redes de Capa 3 subyacentes, básicamente, superposición basada en máquina. No obstante, VXLAN también se puede utilizar como una superposición basada en red ya que brinda ventajas de escalabilidad a los switches Leaf en una arquitectura Fabric o Clos. En los centros de datos, VXLAN es el protocolo más utilizado para crear redes superpuestas. El protocolo VXLAN aborda las necesidades de los centros de datos de múltiples inquilinos al proporcionar la segmentación necesaria a gran escala.

Los servidores dentro de un centro de datos en la nube normalmente presentan envíos TCP/IP, que se encapsulan con información de encabezado de capa 2. En muchos casos, este encabezado también contiene una etiqueta VLAN que el inquilino puede estar usando para identificar diferentes departamentos dentro de su organización. Desde el punto de vista del servidor o de la VM, envía y recibe este tipo de tramas a través de

una red de capa 2 adjunta. El truco aquí es cómo admitir múltiples inquilinos como este en una gran red de centro de datos a nivel de capa 3, y esto es posible gracias a los protocolos de tunelización como VXLAN. El protocolo VXLAN proporciona estas funciones al encapsular estas tramas con una etiqueta VXLAN junto con un encabezado UDP. VXLAN define un esquema de encapsulación MAC-in-UDP en el que se agrega un encabezado VXLAN a la trama de capa 2 original y luego se coloca en un paquete IP UDP. Con esta encapsulación MAC-in-UDP, VXLAN tuneliza la red de Capa 2 sobre la red de Capa 3.

VXLAN utiliza dispositivos de extremo de túnel virtual (VTEP) para asignar dispositivos en segmentos locales a segmentos de VXLAN. Un VTEP realiza el encapsulado y desencapsulado del tráfico de Capa 2 y cada VTEP consta de al menos dos interfaces; una interfaz de switch en el segmento LAN local y una interfaz IP en la red IP de transporte. Debido a que VXLAN se ejecuta en una red IP, la función ECMP de las redes de capa 3 está disponible de manera innata para su uso.

Una vez que el centro de datos comienza a expandirse, podemos ver rápidamente lo difícil que se vuelve administrar VXLAN utilizando el comportamiento de inundación y aprendizaje del plano de control de multicast. Debido a este problema de escala, se creó el plano de control Ethernet VPN (EVPN), utilizando Multi-Protocol BGP (MP-BGP). Esta arquitectura VXLAN sin controlador se basa en que cada nodo es miembro de la superposición de EVPN.

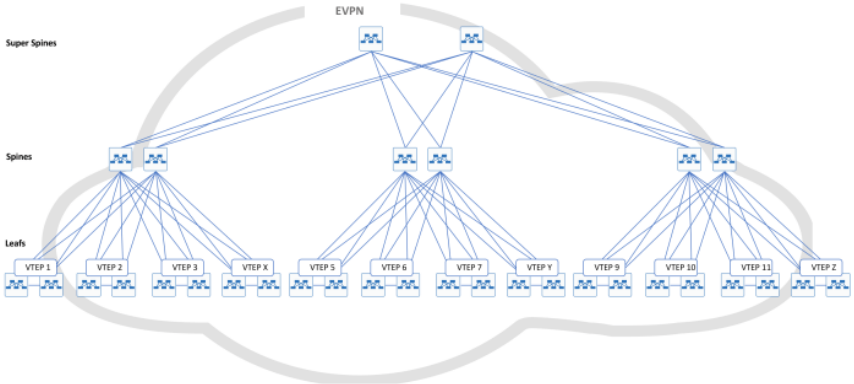


Ilustración 26 – EVPN con VXLAN

- **VXLAN con MP-BGP EVPN:**

EVPN es una extensión basada en estándares RFC 7432[75] para BGP que proporciona un plano de control para VXLAN brindando servicios VPN de Capa 2 y Capa 3.BGP como plano de control para VXLAN nos permite utilizar un solo protocolo de enrutamiento para administrar nuevas capacidades, como el aprendizaje de direcciones MAC y la tenencia múltiple de VRF, al tiempo que proporciona rutas múltiples optimizadas de igual costo (ECMP) a través de los datos intra-DC y externo-DC. En el contexto de un plano de control VXLAN, usamos EVPN como una superposición de virtualización de red (NVO).

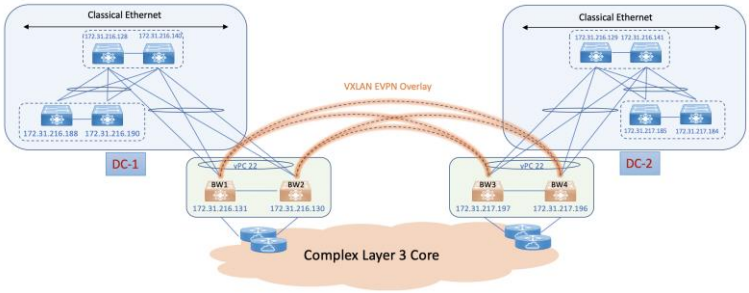


Ilustración 27 – Interconexión VXLAN EVPN

No hace mucho tiempo, era difícil considerar VXLAN para implementar la comunicación entre centros de datos. Ahora, la capacidad de usar MP-BGP EVPN para implementar un plano de control VXLAN lo ha hecho posible. Gracias a MP-BGP EVPN, se resuelve el problema de minimizar la inundación y el aislamiento de los segmentos distribuidos cuando se usa VXLAN.

En general, vale la pena decir que, a juzgar por los documentos de los principales fabricantes, hay grandes esperanzas en la implementación de MP-BGP EVPN. Cisco, por ejemplo, ofrece MP-BGP EVPN como una arquitectura SDN estándar para implementar comunicaciones VXLAN en lugar del Protocolo de administración de base de datos Open vSwitch (OVSDB).

NVGRE:

NVGRE es un protocolo estandarizado del IETF y respaldado a su vez por Microsoft, Intel, HP y Dell que se define en el RFC 7637. [76] La virtualización de red mediante encapsulación de enrutamiento genérico (NVGRE) es similar a su homólogo VXLAN, ya que ambos permiten crear redes superpuestas de Capa 2 sobre redes subyacentes de Capa 3.

El formato de trama NVGRE es muy similar al formato de trama de VXLAN, salvo que en vez de usar un encabezado UDP y un encabezado VXLAN, solo se usa un encabezado GRE, lo que reduce el tamaño del frame en unos pocos bytes. El encabezado GRE contiene el ID de protocolo único (0x6558) para tramas NVGRE, así como un identificador de segmento virtual (VSID) de 24 bits que, al igual que VXLAN, puede admitir hasta 16 millones de subredes de inquilinos únicas. Otra diferencia es que el encabezado de la capa 2 interna no contiene una etiqueta VLAN y, si existe, se elimina antes de que se encapsule la trama. Entonces, en este caso, el VSID también se puede usar para segregar varios segmentos de red virtual para un arrendatario determinado.

NvGRE Ethernet 802.1Q frame - 1588 bytes/octets														
Pre SFD	O-DMAC	O-SMAC	O-TAG 0x0810 0	Type 0x810 0	Outer IP header	NvGRE 0x6558	I-DMAC	I-SMAC	C-TAG 0x810 0	Type 0x080 0	IP 20 bytes	TCP 20 bytes	Payload / Data 6-1460	CRC FCS	IFG
8	6	6	4	2	20	8	6	6	4	2	46-1500			4	12

Ilustración 28 – Frame NvGRE

El Network Virtual Endpoint (NVE) definido en la especificación NVGRE es muy similar al VTEP definido en VXLAN, NVE forma parte del hipervisor Hyper-V de Microsoft.[77]

6.2.3 STT

STT o (*Stateless Transport Tunneling Protocol*) es un protocolo de tunelización para redes superpuestas definido por el IETF dentro del draft-davie-stt-08 [78] en 2012 bajo la colaboración de Nicira. Las tecnologías anteriormente estudiadas como VXLAN y NVGRE tienen el mismo objetivo que STT, proporcionar redes ethernet de capa 2 sobre una infraestructura IP escalable donde la diferencia principal entre ellos radica en el formato de encapsulación y su aproximación al plano de control. STT es particularmente útil cuando algunos puntos finales del túnel están en sistemas finales, ya que utiliza las capacidades de las tarjetas de interfaz de red o NIC para mejorar su rendimiento.

Realizando un repaso de conceptos sobre el uso de TCP en las aplicaciones, estas perciben una conexión TCP como un flujo de bytes confiable. Las aplicaciones envían flujos de bytes a un socket abierto y la pila TCP/IP del sistema operativo divide los datos en paquetes TCP más IP individuales, anteponen el encabezado MAC construido a partir del caché ARP delante del encabezado IP y envía tramas de capa 2 a la NIC para su transmisión. Las NIC típicas pueden segmentar tramas TCP-IP-MAC, pero no pueden segmentar tramas TCP-IP-MAC-VXLAN-UDP-IP-MAC o tramas TCP-IP-MAC-NVGRE-IP-MAC. El envío de tramas de capa 2 a través de VXLAN o NVGRE inhabilita la descarga de TCP en la mayoría de las NIC de servidor disponibles en la actualidad y por este motivo el protocolo STT resulta de utilidad.

STT usa un encabezado que se parece al encabezado TCP de una NIC. Por lo que la NIC puede realizar una descarga de segmento grande en lo que cree que es un datagrama TCP.Sin embargo, lo que se está manejando en la NIC es un marco TCP-IP-MAC de gran tamaño hasta 64 bits con encabezado STT-IP-MAC. Los segmentos TCP producidos por la NIC no son los segmentos TCP reales, sino segmentos de la trama STT que se pasan a la NIC.

STT Ethernet 802.1Q frame - 1608 bytes/octets																
Pre SFD	O-DMAC	O-SMAC	O-TAG 0x0800	Type 0x8100	Outer IP Header	Outer TCP Header	STT Header	I-DMAC	I-SMAC	C-TAG 0x8100	Type 0x0800	IP 20 bytes	TCP 20 bytes	Payload / Data 6-1460		CRC FCS	IFG
8	6	6	4	2	20	20	8	6	6	4	2	46-1500				4	12

Ilustración 29 – Frame STT

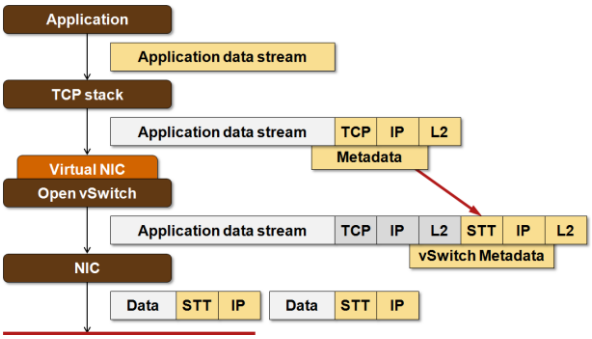


Ilustración 30 – NIC virtual STT

Como se puede apreciar STT, es de gran importancia dentro del segmento de la máquina virtual que proporciona la superposición. No obstante, fuera de este dominio STT presenta dificultades operativas. Un paquete STT aparenta ser un mismo paquete TCP, pero el procesamiento de un paquete STT en la recepción es completamente diferente de TCP, no hay protocolo de enlace de tres vías para establecer una conexión.

Por lo tanto, si utilizamos el protocolo STT en un equipo final que forma parte del túnel, se tendrá que configurar previamente este equipo para poder realizar el procesamiento del paquete TCP de forma correcta. Del mismo modo, como tampoco existe el control de congestión en STT. Los paquetes STT que se envían deberán permanecer dentro del perímetro de la superposición gestionada por el controlador virtual que se esté utilizando, ya que un envío de un paquete STT fuera de este perímetro provocaría la incomunicación de la red de superposición desplegada en cualquier topología.

6.2.4 GENEVE

GENEVE es un protocolo de tunelización para redes superpuestas definido por el IETF bajo el estándar RFC8926.[79] Repasando los protocolos anteriores, vemos que los tres encapsulan los datos de la aplicación en un nuevo campo de encabezado fijo más grande. Ese tamaño de encabezado es de 24 bits para VXLAN y NVGRE y 64 bits para STT. Aunque la utilización de estos métodos de tunelización de encapsulación no requiere de ningún cambio en la infraestructura del hardware, ambos protocolos presentan un problema en común y es que no son compatibles entre sí.

El principal objetivo del protocolo GENEVE es definir un protocolo único de encapsulación que, a diferencia de los protocolos anteriores, no incluya ninguna especificación para el plano de control. En GENEVE, el plano de datos es genérico y lo suficientemente extensible para admitir planos de control actuales y futuros. Los componentes del túnel se pueden implementar de manera eficiente tanto en hardware como en software sin restringir las capacidades y se mantiene el alto rendimiento sobre las estructuras IP existentes. Su uso se aplica a entornos de centros de datos con arquitectura Clos.
Los paquetes encapsulados de GENEVE están diseñados para transmitirse a través de equipos de red estándar. Los paquetes se envían desde un punto final de túnel o (TEP) a uno o más puntos finales de túnel utilizando direccionamiento de unidifusión o multidifusión. La aplicación cliente y el host en el que se ejecuta no se modifican de ninguna manera. Las aplicaciones generan paquetes IP idénticos como si se estuvieran comunicando a través de conmutadores y enrutadores de hardware. La dirección IP de destino incluida en el paquete es significativa solo dentro de la red virtual del inquilino de la nube. El extremo del túnel luego encapsula el paquete IP del usuario final en el encabezado GENEVE, agregando el identificador del túnel que especifica la red virtual del arrendatario seguido de cualquier opción. El encabezado consta de campos que especifican que se trata de un paquete GENEVE, la longitud total de las opciones, si las hay, el identificador del túnel y la serie de opciones. Luego, el paquete completo se transmite al punto final de destino en un paquete UDP estándar que se admite a través de IPv4 e IPv6. El extremo del túnel receptor elimina el encabezado, interpreta las opciones incluidas y dirige el paquete del usuario final a su destino dentro de la red virtual indicada por el identificador del túnel.

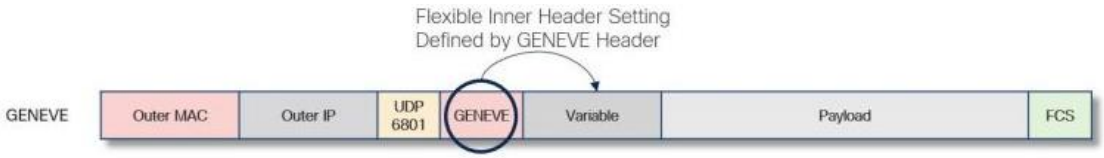


Ilustración 31 – Frame GENEVE

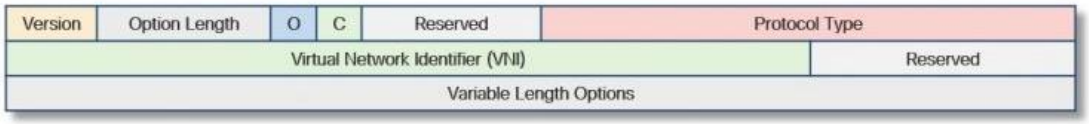


Ilustración 32 – Cabecera Variable GENEVE

Los extremos del túnel GENEVE solo se comunicarán entre sí y la infraestructura de la red manejará los paquetes de manera idéntica a cualquier otro paquete UDP. El formato de datos es compatible con todas las capacidades de VXLAN, NVGRE y STT.

GENEVE es el protocolo de tunelización predeterminado para OVN (Open Virtual Network), que se promociona a su vez como una implementación de OVS (OpenvSwitch) y cuenta con el respaldado de los principales fabricantes VMware, Red Hat, Intel y Microsoft.

6.2.5 OTV y LISP

Todos los protocolos anteriormente analizados (a excepción de VXLAN con EVPN), son protocolos cuyo ámbito de aplicación está dentro del propio sistema del CPD o intra-DC.

Los nuevos MSDC no solo requieren de conectividad intra-DC, también tienen que dar accesibilidad de conexión a máquinas y dispositivos desplegados en diferentes ubicaciones o entre sistemas de múltiples MSDC. Actualmente, en el mercado existen dos protocolos propietarios de Cisco System que cubren estas necesidades.

OTV:

OTV o virtualización de transporte superpuesta, es un protocolo de tunelización para redes superpuestas definido por el IETF bajo el estándar 802.1Q [80]en colaboración con Cisco System, que se creó para ampliar un dominio de transmisión entre dos o más centros de datos a través de las implementaciones de VPN a nivel de capa 2 sobre una red IP. OTV implementa el concepto de enrutamiento en direcciones MAC de la siguiente forma. Los dispositivos que ejecutan tecnología OTV sobre el protocolo de enrutamiento dinámico IS-IS intercambian datos de sus direcciones MAC para conocer los equipos finales que cuelgan de ellos. Esto permite que cada dispositivo OTV tenga una tabla de enrutamiento de direcciones MAC común. Cuando reciben un paquete cuya dirección MAC de destino es un equipo ubicado en algún segmento remoto de la red de capa 2, el dispositivo OTV determina a partir de su tabla dónde reenviar el paquete. El encabezado se agrega al paquete y se envía primero al dispositivo OTV remoto y luego al equipo de destino. Los paquetes ethernet que se transfieren entre dispositivos OTV, se encapsulan en datagramas UDP.

OTV Ethernet 802.1Q frame by IETF - 1596 bytes/octets																
Pre SFD	O-DMAC	O-SMAC	O-TAG 0x0800	Type 0x8100	Outer IP header	O-UDP Header	OTV Header	I-DMAC	I-SMAC	C-TAG 0x8100	Type 0x0800	IP 20 bytes	TCP 20 bytes	Payload / Data 6-1460	CRC FCS	IFG
8	6	6	4	2	20	8	8	6	6	4	2	46-1500			4	12

Ilustración 33 – Frame OTV

En general, OTV es muy similar a la lógica de VXLAN. Pero a diferencia de VXLAN, la tecnología OTV está diseñada principalmente para la conexión de redes de capa 2 entre centros de datos. OTV utiliza IS-IS para el plano de control al igual que TRILL/FabricPath.

OTV admite dos modos de funcionamiento para garantizar una mayor comunicación entre dispositivos. En el primer caso, el tráfico multicast se utiliza para comunicar dispositivos OTV entre sí. Todos los dispositivos OTV indican a la red que quieren recibir tráfico multicast dirigido a un grupo específico, a través del protocolo IGMP. Estos mensajes son enviados por la red a todos los dispositivos OTV. Este comportamiento es muy similar al protocolo de enrutamiento dinámico normal. No obstante, solo los dispositivos OTV pueden estar en diferentes subredes. Para que todo esto funcione, la red debe soportar la transmisión de tráfico multicast, incluido su enrutamiento algo que no resulta fácil si existe tráfico Inet.

En el caso de utilizar tráfico unicast para la interacción entre dispositivos OTV, uno de ellos se configura como servidor de adyacencia, en todos los demás dispositivos OTV, se indica su dirección para poder establecer una conexión con el servidor. Este servidor, es el encargado de recopilar los datos sobre todos los dispositivos OTV y distribuirlos a los demás. Habiendo recibido esta información, cada dispositivo OTV puede comunicarse con otros utilizando paquetes unicast. Sin embargo, el intercambio de tablas de direcciones MAC es más pesado en OTV en comparación con el generado por el tráfico multicast. Para reducir la influencia de los segmentos de capa 2 distribuidos entre sí y optimizar el tráfico transmitido entre ellos, OTV proporciona un aislamiento parcial que consta de las siguientes características:

- No se transmiten paquetes BPDU de protocolos STP entre dispositivos OTV.
- Cada segmento construye su propia topología STP independiente.
- El tráfico de unicast desconocido no se transmite entre dispositivos OTV. Esta lógica de trabajo parte de la suposición de que cualquier dispositivo, tarde o temprano, transmitirá algunos datos y descubriremos su dirección MAC.
- Transmisión optimizada de mensajes ARP. En los dispositivos OTV, todas las respuestas ARP provenientes de hosts remotos se almacenan en caché. Por lo tanto, el dispositivo OTV local puede responder a una solicitud ARP si alguien más ya envió una similar.

Como se señaló anteriormente, OTV, a diferencia de VXLAN, proporciona comunicación solo entre centros de datos. De hecho, VXLAN es un protocolo más universal. Sin embargo, los requisitos de hardware de OTV son ligeramente inferiores. Cisco suele recomendar OTV en diseños entre centros de datos, donde se utiliza TRILL/FabricPath en ámbito intra-DC.

LISP:

La creación de LISP fue inicialmente motivada por discusiones durante el Taller de enrutamiento y direccionamiento patrocinado por IAB que se llevó a cabo en Ámsterdam en octubre de 2006 (RFC 4984). [81] Una conclusión clave del taller fue que el sistema de direccionamiento y enrutamiento de Internet no estaba escalando bien frente al crecimiento explosivo de los MSDC. Una de las razones de este escalamiento deficiente fue el número cada vez mayor de sitios multiproveedor que no podían abordarse como parte de prefijos agregados basados en topología o proveedores. En la arquitectura actual de enrutamiento y direccionamiento de Internet, la dirección IP de un dispositivo se usa como un único espacio de nombres que expresa simultáneamente dos funciones de un dispositivo: su identidad y cómo está conectado a la red. Cuando ese dispositivo se mueve, debe obtener una nueva dirección IP tanto para su identidad como para su ubicación.
LISP o Localizador de Cisco/Protocolo de separación de ID, es un protocolo definido dentro del RFC 6830 que permite la separación de la identidad del punto final y su ubicación. Es un protocolo propietario de Cisco al igual que OTV, pero cuya misión principal es solventar el problema de enrutar correctamente el tráfico si una VM se ha movido, por ejemplo, de un centro de datos a otro.

Cuando un equipo se migra a otra ubicación, es difícil redirigir ese tráfico ya que parte de su configuración de redes IP o subredes son distintas. Para solucionar este inconveniente, una de las soluciones propuestas es la separación del identificador del servidor en dos partes:

- **Direcciones de identificador de punto final (EID):** Son direcciones IP y prefijos que identifican los puntos finales. La accesibilidad de EID en los sitios LISP se logra mediante la resolución de asignaciones de EID a RLOC.

- **Direcciones de localizador de enrutamiento (RLOC):** Son direcciones IP y prefijos que identifican los diferentes enrutadores en la red IP. La accesibilidad dentro del espacio RLOC se logra mediante métodos de enrutamiento tradicionales.

La dirección del (EID) y la dirección (RLOC) en caso de una migración de una máquina virtual a otro centro de datos, mediante LISP, se simplifica con que la dirección IP del dispositivo (EID) seguirá siendo la misma en ambas ubicaciones y solo cambiaría la ID de la ubicación.

En los equipos de Cisco, se ha generalizado bastante una combinación de tecnologías FabricPath dentro del centro de datos y OTV entre centros de datos. Estas tecnologías están bien desarrolladas, son fáciles de configurar y la mayoría de los diseños de Cisco se basan en ellas. Hay ejemplos de centros de datos donde se usa esto. Es cierto que vale la pena señalar que su uso requiere un hardware no tan simple y económico. LISP tiene una limitación en el sentido de que solo admite la superposición de Capa 3. No puede transportar la dirección MAC porque descarta el encabezado Ethernet de capa 2. En ciertas tecnologías resulta necesaria la dirección MAC por lo que en tales casos se recomienda implementar VXLAN.
La tecnología LISP se destaca un poco, resolviendo el problema de la transferencia óptima de tráfico hacia el servidor y, en algunas tareas, hacia el cliente. Por lo tanto, se recomienda utilizar LISP independientemente de las tecnologías superpuestas entre los

centros de datos. LISP no es la solución fácil desde el punto de vista de la implementación al problema de la transferencia de tráfico no óptima. Por ejemplo, para resolver completamente el problema del tráfico no óptimo del cliente al servidor en el centro de datos, es decir, de Inet a DC, se debería configurar LISP para cada cliente remoto, lo que resulta ser una tarea bastante ardua y complicada.

6.2 Comparativa de Tecnologías

	VXLAN	NVGRE	STT	GENEVE	OTV	LISP
Tipo de Superposición	Overlay VM	Overlay VM	Overlay VM	Overlay VM	Overlay Network	Overlay Network
Estándar	RFC 7348	RFC 7637	draft-davie-stt-08	RFC 8926	802.1Q	RFC 6830
Tipo de Comunicación	Intra-DC	Intra-DC	Intra-DC	Intra-DC	Externa-DC	Externa-DC
Identificación de Superposición	24 bits (VNI)	24 bits (VSID)	64 bit (ID)	8 - 128 bits	24 bits (ID)	24 bits (LISP-ID)
Cabecera overlay	50 bytes	42 bytes	76 bytes	58 bytes	42 bytes	50 bytes
IS-IS	No	No	No	No	Sí	Sí
NIC	No	No	Sí	Sí	No	No
OpenSource	OpenvSwitch	No	Open vSwitch	Open vSwitch	No	Cisco
Escalabilidad	16 millones	16 millones	Limitado a 1000 equipos	16 millones	No aplica	No aplica
Fabricante	Cisco, Juniper, Aruba, Arista, Vmware, Brocade.etc	Microsoft, Arista, Huawei, Dell y HP	Vmware, Broadcom	RedHat, Vmware	Cisco	Cisco

Tabla IX – Comparativa de tecnologías Overlay (elaboración propia)

7. LAB 1: Despliegue de una arquitectura Clos con controlador SDN

7.1 Elección de escenario a desplegar

En todo proyecto, resulta casi imprescindible aportar una simulación práctica que sostenga los fundamentos teóricos del paradigma SDN analizados anteriormente. Por lo que es este apartado se desplegará una topología Spine-Leaf bajo el control de un controlador SDN puro como es el caso de ONOS. Para la gestión de los swiches y despliegue de la topología se utilizará la herramienta Mininet. Se ha optado por una opción opensource que permita la desligación de cualquier software fabricante como pilar base de la filosofía SDN.Los análisis de paquetes Openflow se realizarán a través

de la herramienta Wireshark e Iperf de Mininet que nos permite conocer el consumo de ancho de banda de los enlaces. A continuación, se exponen las opciones que se han considerado para crear el escenario virtual de un controlador SDN.

Opción 1: Máquinas virtuales separadas para la instalación de Mininet y Onos de forma independiente pero conectadas entre sí.
Opción 2: Instalación de Mininet y Onos en una sola VM con un 1 nodo de gestión.
Opción 3: Instalación de Mininet y Onos en una sola VM con 3 nodos de gestión bajo configuración previa de Dockers para formar un clúster de controladores.

Para el montaje de un laboratorio SDN dentro de la VM se han estudiado varias opciones, pero se ha seleccionado finalmente la opción 3 que permite el balanceo de carga entre controladores algo que se considera muy útil cuando el número de equipos y enlaces es alto. De esto modo, ante la caída de un enlace o equipo que forma la topología siempre existirá un controlador redundante que sigue realizando la operación.

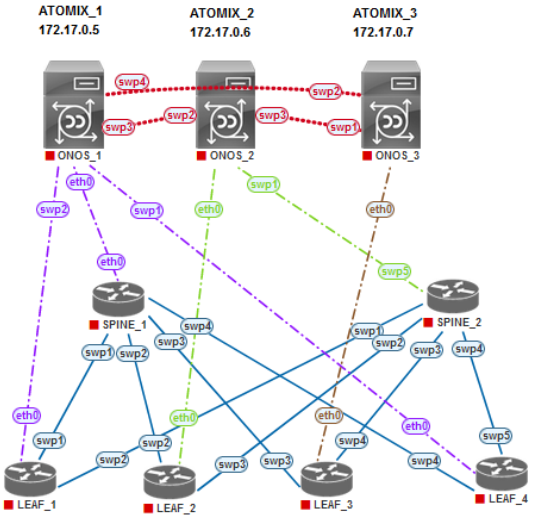


Ilustración 34 - Clúster de controladores

7.2 Análisis e implementación

Para poder realizar el TFG correctamente, es necesaria la instalación de un sistema operativo GNU/Linux funcional. En este caso se ha elegido la distribución GNU/Linux Kubuntu 16.04 LTS, aunque cualquier otra distribución es válida. Su despliegue se realizará dentro de un sistema virtualizado en VMware Workstation, pero podría utilizarse VirtualBox de igual forma. Para poder desplegar la virtualización, se han creado dos interfaces VMnet1 (Host-only) con el direccionamiento 192.168.247.0/24 y VMnet8 con NAT y direccionamiento 192.168.91.0/24. El direccionamiento para el clúster de Dockers es el asignado por defecto 172.17.0.0/16.

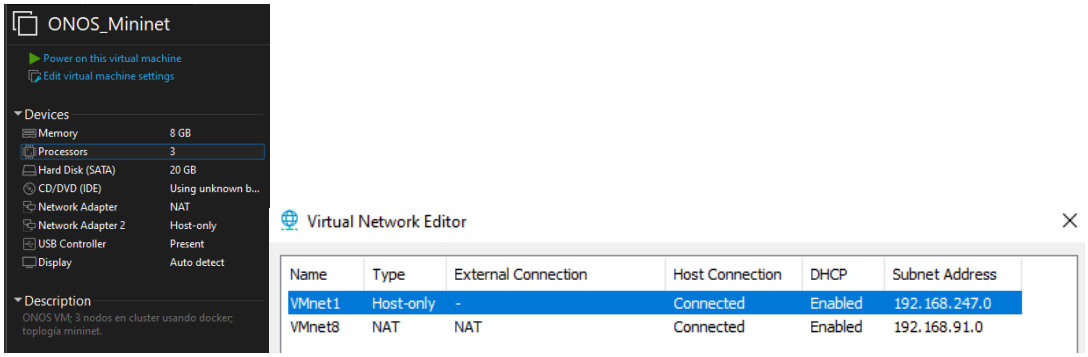


Ilustración 35 - Configuración VM

Las interfaces en la VM quedan configuradas de la siguiente forma:

```
sdn@bcastroseSDN-ONOS:~$ ifconfig
docker0   Link encap:Ethernet  HWaddr 02:42:2c:b9:5e:e1
          inet addr:172.17.0.1  Bcast:172.17.255.255  Mask:255.255.0.0
          inet6 addr: fe80::42:2cff:feb9:5ee1/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2297 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2341 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:91532 (91.5 KB)  TX bytes:174810 (174.8 KB)

ens33     Link encap:Ethernet  HWaddr 00:0c:29:1d:0c:e9
          inet addr:192.168.91.136  Bcast:192.168.91.255  Mask:255.255.255.0
          inet6 addr: fe80::f5f5:db9c:5178:97b0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1030 errors:0 dropped:0 overruns:0 frame:0
          TX packets:612 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1313806 (1.3 MB)  TX bytes:47507 (47.5 KB)

ens34     Link encap:Ethernet  HWaddr 00:0c:29:1d:0c:f3
          inet addr:192.168.247.128  Bcast:192.168.247.255  Mask:255.255.255.0
          inet6 addr: fe80::c851:d30c:6e44:4418/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:5 errors:0 dropped:0 overruns:0 frame:0
          TX packets:52 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:866 (866.0 B)  TX bytes:6776 (6.7 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:98 errors:0 dropped:0 overruns:0 frame:0
          TX packets:98 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:9251 (9.2 KB)  TX bytes:9251 (9.2 KB)
```

Ilustración 36 - Interfaces VM

7.2.1 MININET

El emulador Mininet es una plataforma de pruebas de red de código abierto y la herramienta de apoyo a la investigación de las redes SDN con OpenFlow más conocida. Mininet utiliza hosts virtuales, switches y enlaces para crear una red en un solo núcleo del sistema operativo. Las aplicaciones de red basadas en Unix/Linux, se pueden ejecutar en los hosts virtuales desplegados desde Mininet.

La instalación de Mininet se puede realizar de forma nativa dentro del SO Kubuntu 16.04 LTS, esta versión es la recomendada junto con la 18.08 LTS por parte del proyecto ONOS. Para poder realizar dicha instalación es necesario tener acceso a internet desde la VM. De este modo se pueden instalar diversos elementos como Wireshark junto con los otros elementos clave del emulador. Las instrucciones para la instalación de Mininet están disponibles en su página web y son descritas en el Anexo 12.2.1 . En el momento de realizar la memoria la versión instalada de Mininet es la 2.3.0d1.

Después de instalar el emulador, sólo con escribir la instrucción ***$ sudo mn*** en el Shell se ejecuta Mininet y se crea una topología denominada mininal. Aquí el prompt cambia a ***'mininet>'*** lo que indica que se está dentro del entorno.

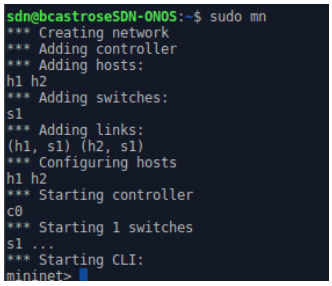


Ilustración 37 - Topología básica

La instrucción ***sudo mn --help*** nos permite utilizar diferentes parámetros, entre estos algunos de ayuda, para limpiar y reiniciar el emulador y otros para ejecutar topologías de otros tipos como minimal, single, linear y tree además de permitir la creación de topologías personalizadas.

Para el estudio del escenario a desplegar se ha utilizado una topología personalizada en modo de árbol o Tree con 3 niveles de profundidad. Para ejecutarla de forma manual se puede utilizar la instrucción sudo ***mn --topo tree, depth=n*** donde n es la profundidad en niveles de la red. El uso de topologías personalizadas debe implementarse mediante scripts utilizando las librerías de Python y la API de Mininet, otra opción es el uso de la interfaz gráfica que proporciona Mininet por defecto y que se denomina Miniedit.Para ejecutarla debemos ir al directorio /mininet/mininet/examples/miniedit.py y lanzar la aplicación a través de python3 previamente definido.

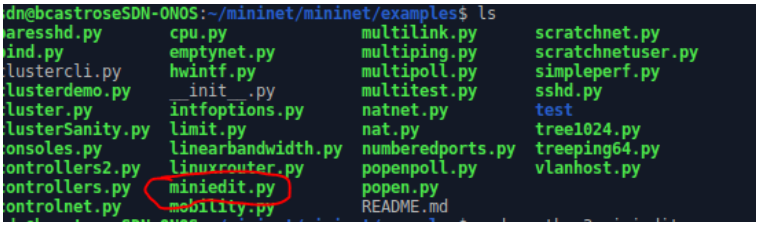


Ilustración 38 - Miniedit

Para este caso, se ha utilizado la herramienta gráfica Miniedit, con el despliegue del siguiente escenario que consta de:

Controladores
C1: 172.17.0.5
C2: 172.17.0.6
C3: 172.17.0.7

Switches	
S1: 172.17.0.6	S11: 172.17.0.6
S2: 172.17.0.7	S12: 172.17.0.5

	S13: 172.17.0.6
	S14: 172.17.0.6
	S15: 172.17.0.6

Los switches Spine y Leaf al estar conectados a los controladores a través del protocolo OpenFlow no necesitan de una IP predefinida. Adquirirán la Ip del controlador SDN al que estén conectados.

Hots			
H11: 10.0.0.1	H21: 10.0.0.6	H31: 10.0.0.11	H41: 10.0.0.16
H12: 10.0.0.2	H22: 10.0.0.7	H32: 10.0.0.12	H42: 10.0.0.17
H13: 10.0.0.3	H23: 10.0.0.8	H33: 10.0.0.13	H43: 10.0.0.18
H14: 10.0.0.4	H24: 10.0.0.9	H34: 10.0.0.14	H44: 10.0.0.19
H15: 10.0.0.5	H25: 10.0.0.10	H35: 10.0.0.15	H45: 10.0.0.20

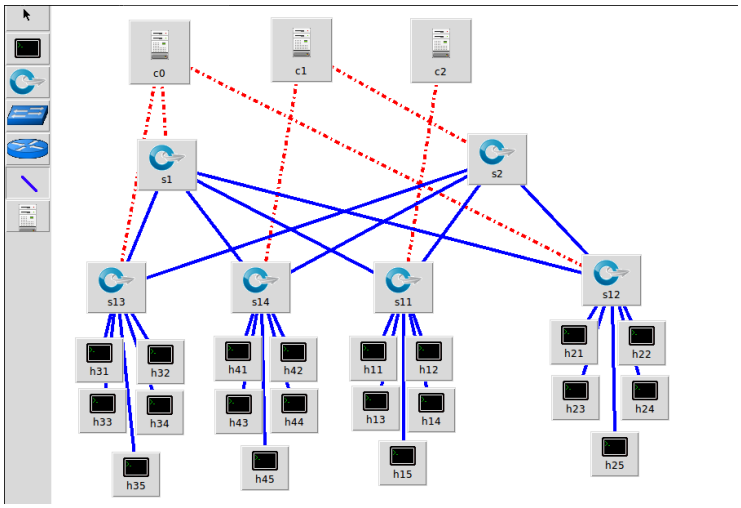
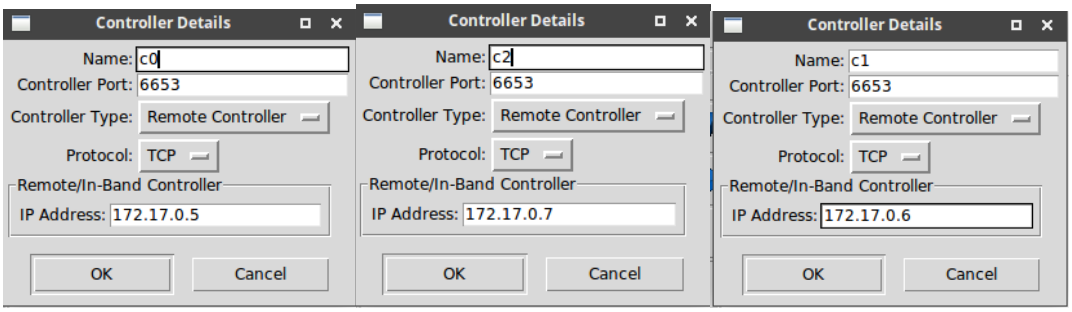


Ilustración 39 - Despliegue Topología Clos (elaboración propia)

Entre las características que podemos configurar dentro de la interfaz gráfica de Mininedit están, la versión del protocolo OpenFlow (1.3), el tipo de switch a desplegar (OpenvSwitch) y la red IP (10.0.0.0/8) está última es la definida por defecto. Otras características adicionales son sFlow y NetFlow dentro de las opciones de gestión de tráfico en OpenvSwitch. En la definición del tipo de controlador se elige "Remote Controller.



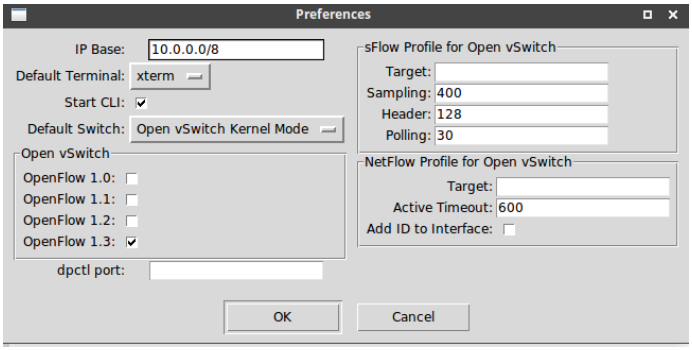


Ilustración 40 - Configuraciones Controladores y red

Una vez definidos los parámetros de los equipos a desplegar y presentada la topología se carga el script para ejecutarse en Mininet. ***File/Export Level 2 Script/Lab1.py.***El resultado del script es la creación de las conexiones y los dispositivos según la topología personalizada, tal y como se puede observar en la Ilustración.

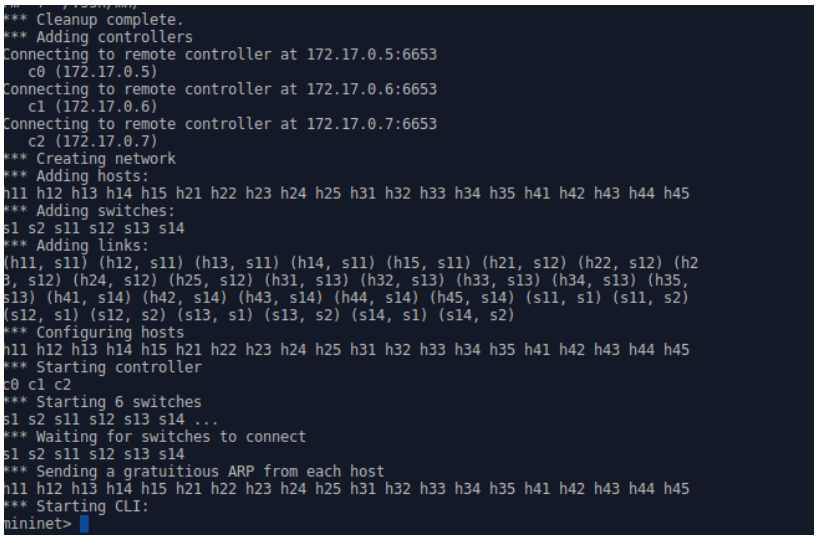


Ilustración 41 - Ejecución topología Clos en Mininet

7.2.2 ONOS

ONOS proporciona el plano de control para una red definida por software (SDN), gestiona componentes de red, como switches y enlaces, y ejecuta programas o módulos de software para proporcionar servicios de comunicación a hosts finales y redes vecinas. El beneficio más importante de ONOS es que proporciona una plataforma útil y utilizable para SDN. Las aplicaciones y casos de uso de ONOS a menudo consisten en servicios personalizados de enrutamiento, administración o monitoreo de comunicaciones. El núcleo y los servicios centrales de ONOS, así como las aplicaciones de ONOS, están escritos en Java y se cargan en el contenedor Karaf OSGi. OSGi es un sistema de componentes para Java que permite que los módulos se instalen y ejecuten dinámicamente en una sola Java Virtual Machine.

La instalación de ONOS necesita de unos requisitos previos de software para su funcionamiento:

- Instalación Java 11→ Ver Anexo 12.2.2

- Instalación Maven → Ver Anexo 12.2.3
- Instalación Karaf → Ver Anexo 12.2.4
- Instalación Git Core → Ver Anexo12.2.5
- Instalación Curl → Ver Anexo 12.2.6
- Instalación Docker y Atomix → Ver Anexo 12.2.7

Del mismo modo, ONOS requiere que los siguientes puertos estén abiertos para que las funcionalidades correspondientes estén disponibles:

- 8181 para REST API y GUI
- 8101 para acceder a la CLI de ONOS
- 9876 para comunicación dentro del clúster
- 6653 opcional, para OpenFlow
- 6640 opcional, para OVSDB

Los scripts utilizados para ejecutar ONOS como un servicio (utilizados después del proceso de instalación) requieren un usuario especial sin privilegios (a menudo el usuario 'sdn u onos') configurado en el sistema.

Una vez finalizada la instalación previa de sus componentes y del propio SO de ONOS descrito en el Anexo 12.2.9 . Se puede lanzar el servicio, parar, reiniciar o comprobar el estado con el siguiente comando. ***/onos-service [start/stop/restart/status].***

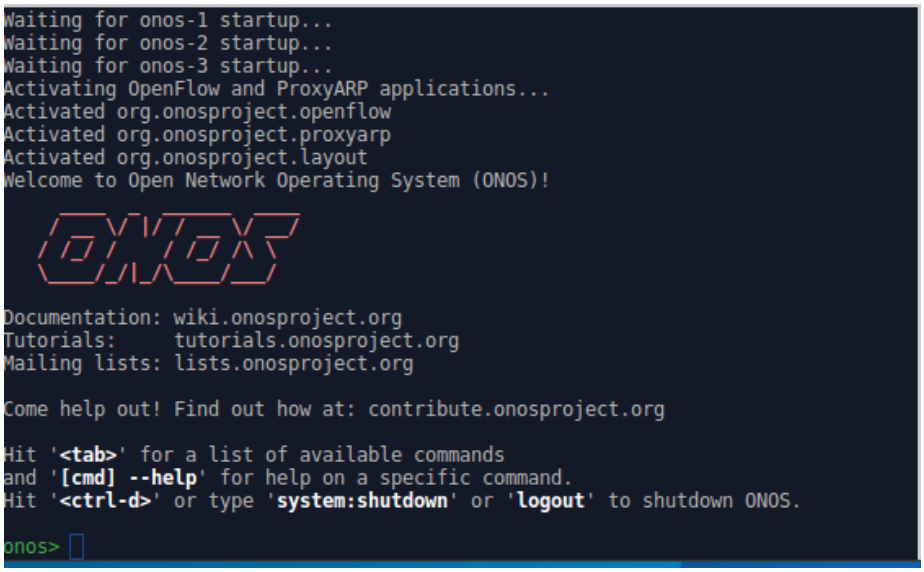


Ilustración 42 - ONOS CLI

Del mismo modo, se puede acceder por Web desde cualquier navegado a través de:

https://172.17.0.X:8181/onos/ui/index.html

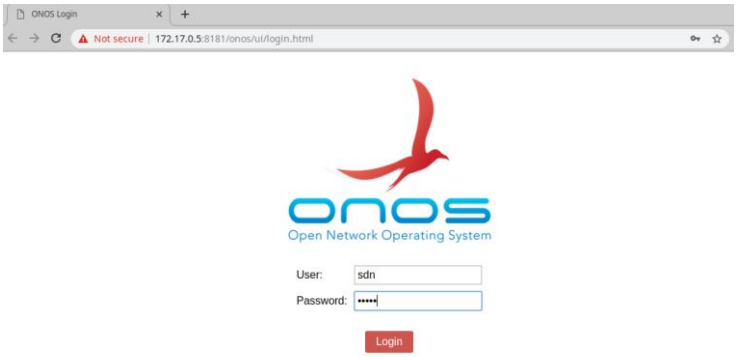


Ilustración 43 - ONOS Web

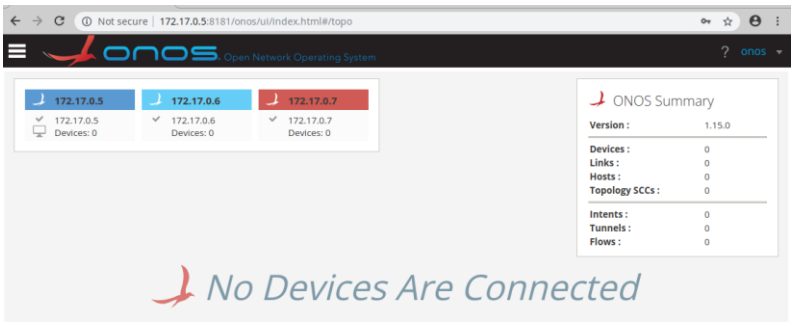


Ilustración 44 - Cluster ONOS

Una vez finalizado y ejecutado el servicio del SO de ONOS hay que ejecutar la topología anteriormente desplegada y personalizada de Miniedit. Para esto,solo necesitamos volver a lanzar el sriptc de Pyton denominado ***Lab1.py*** y automáticamente se realizará el despliegue dentro de ONOS tal y como se puede observar en la imagen siguiente.

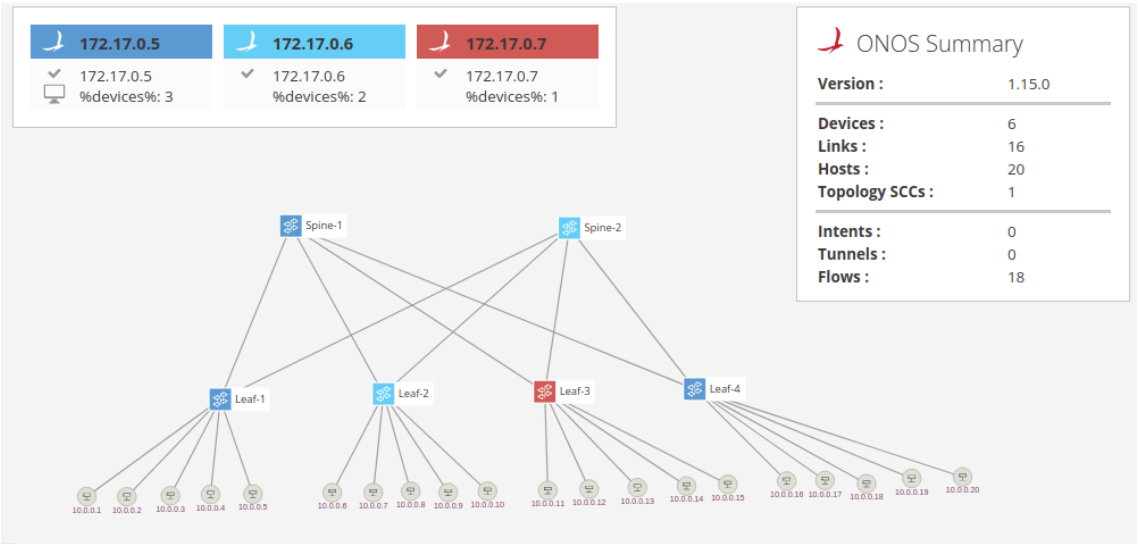


Ilustración 45 - Topología exportada de Mininet a ONOS

A continuación, se realiza un testing sobre los equipos que se han desplegado. Tanto en modo CLI como en modo WEB se pueden consultar los switches y Hosts desplegados.

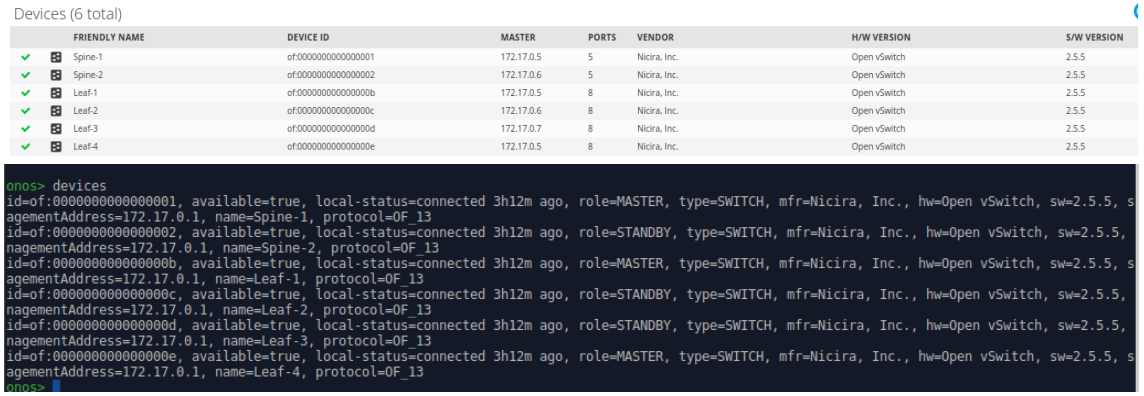


Ilustración 46 - Inventario de equipos en ONOS

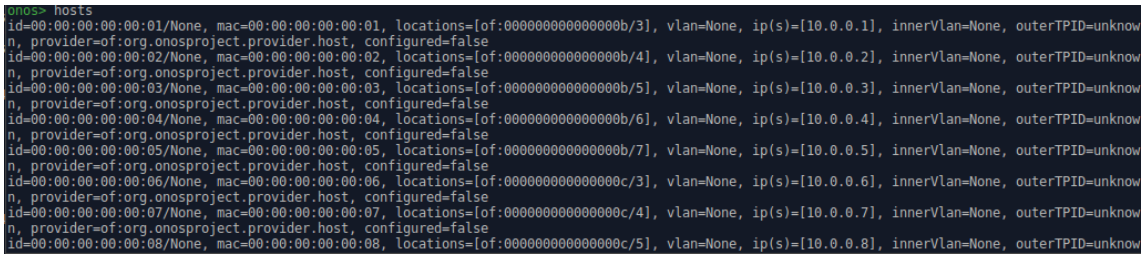


Ilustración 47 - Inventario de hosts

De manera similar ONOS nos permite consultar el estado de los enlaces, el resultado muestra la lista de enlaces descubiertos. Los enlaces se formatean por par de puerto de dispositivo de origen a par de puerto de dispositivo de destino. El campo de "TYPE" indica si el enlace es una conexión directa entre dos dispositivos o no.

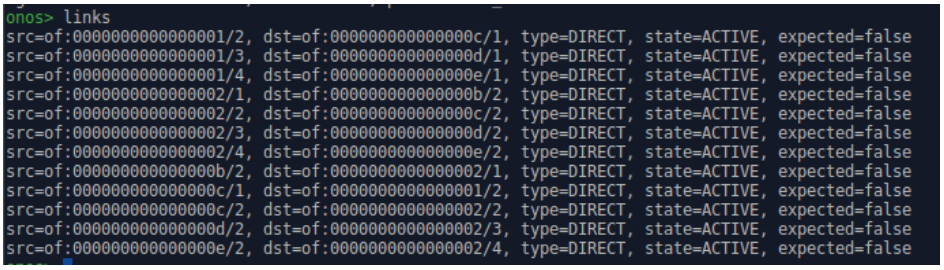


Ilustración 48 - Enlaces desplegados

ONOS utiliza APIs preconfiguradas para el despliegue de servicios adicionales. Para consultar las apps que están por defecto configuradas, basta con introducir el comando ***app –a –s*** o en modo web consultar "Aplicaciones".

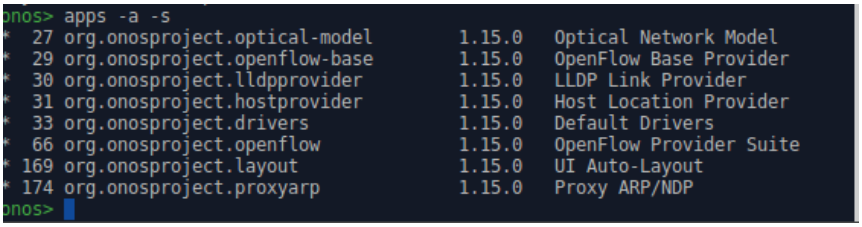


Ilustración 49 - Consulta de Apps instaladas

Un dato curioso que se detecta en la primera ejecución de una topología es que, ONOS no tiene activado por defecto el reenvío reactivo, esto se traduce, a que cuando se quiere lanzar un ping de un equipo a otro, el resultado es una pérdida de paquetes del 100%. Y puede dar lugar a confusión sobre el funcionamiento del sistema. Para corregir este problema, hay que instalar la App "***org.onosproject.fwd***" o definirla previamente en el momento de instalación del SO ONOS.

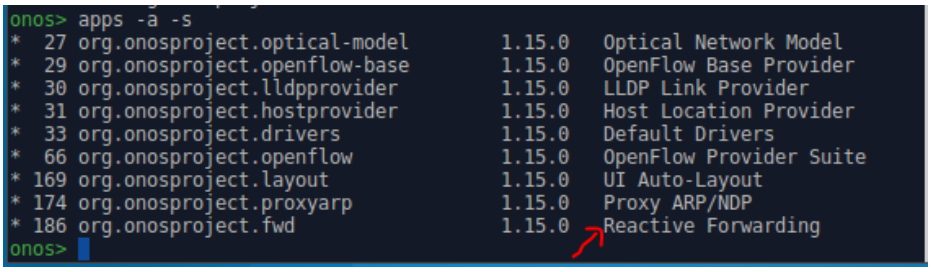


Ilustración 50 - Instalación de App Reenvío reactivo

Una vez instalada, ya podemos realizar ping entre hosts

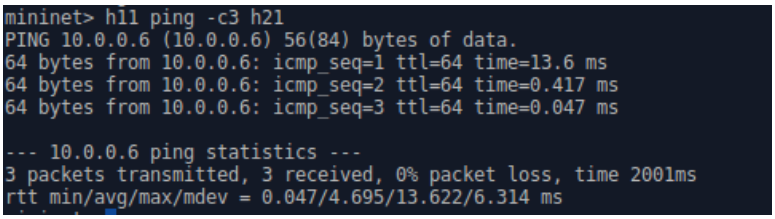


Ilustración 51 - Ejemplo de ping

En ONOS los flujos de datos entre equipos permiten observar previamente en qué estado se encuentran.

- **PENDING_ADD:** El flujo se ha enviado y reenviado al switch.
- **ADDED:** El flujo se agregó al switch.
- **PENDING_REMOVE:** La solicitud para eliminar el flujo se envió y se reenvió al switch.
- **REMOVED:** La regla se eliminó.

Previamente, en la realización de ping desde el equipo "H11" al equipo "H21" habíamos creado un flujo. Ahora podemos consultar su estado con el comando ***flows -s*** en CLI o por WEB pinchando en el equipo.

```
onos> flows -s
deviceId=of:0000000000000001, flowRuleCount=5
    ADDED, bytes=216840, packets=2780, table=0, priority=40000, selector=[ETH_TYPE:bddp], treatment=[immediate=[OUTPUT:CONTROLLER], clearDeferred]
    ADDED, bytes=217074, packets=2783, table=0, priority=40000, selector=[ETH_TYPE:lldp], treatment=[immediate=[OUTPUT:CONTROLLER], clearDeferred]
    ADDED, bytes=0, packets=0, table=0, priority=40000, selector=[ETH_TYPE:arp], treatment=[immediate=[OUTPUT:CONTROLLER], clearDeferred]
    ADDED, bytes=0, packets=0, table=0, priority=5, selector=[ETH_TYPE:arp], treatment=[immediate=[OUTPUT:CONTROLLER], clearDeferred]
    ADDED, bytes=392, packets=4, table=0, priority=5, selector=[ETH_TYPE:ipv4], treatment=[immediate=[OUTPUT:CONTROLLER], clearDeferred]
deviceId=of:0000000000000002, flowRuleCount=5
    ADDED, bytes=216840, packets=2780, table=0, priority=40000, selector=[ETH_TYPE:lldp], treatment=[immediate=[OUTPUT:CONTROLLER], clearDeferred]
    ADDED, bytes=0, packets=0, table=0, priority=40000, selector=[ETH_TYPE:arp], treatment=[immediate=[OUTPUT:CONTROLLER], clearDeferred]
    ADDED, bytes=216840, packets=2780, table=0, priority=40000, selector=[ETH_TYPE:bddp], treatment=[immediate=[OUTPUT:CONTROLLER], clearDeferred]
    ADDED, bytes=0, packets=0, table=0, priority=5, selector=[ETH_TYPE:ipv4], treatment=[immediate=[OUTPUT:CONTROLLER], clearDeferred]
    ADDED, bytes=0, packets=0, table=0, priority=5, selector=[ETH_TYPE:arp], treatment=[immediate=[OUTPUT:CONTROLLER], clearDeferred]
deviceId=of:000000000000000b, flowRuleCount=5
    ADDED, bytes=108420, packets=1390, table=0, priority=40000, selector=[ETH_TYPE:bddp], treatment=[immediate=[OUTPUT:CONTROLLER], clearDeferred]
    ADDED, bytes=108576, packets=1392, table=0, priority=40000, selector=[ETH_TYPE:lldp], treatment=[immediate=[OUTPUT:CONTROLLER], clearDeferred]
    ADDED, bytes=504, packets=12, table=0, priority=40000, selector=[ETH_TYPE:arp], treatment=[immediate=[OUTPUT:CONTROLLER], clearDeferred]
    ADDED, bytes=392, packets=4, table=0, priority=5, selector=[ETH_TYPE:ipv4], treatment=[immediate=[OUTPUT:CONTROLLER], clearDeferred]
    ADDED, bytes=0, packets=0, table=0, priority=5, selector=[ETH_TYPE:arp], treatment=[immediate=[OUTPUT:CONTROLLER], clearDeferred]
deviceId=of:000000000000000c, flowRuleCount=5
    ADDED, bytes=108420, packets=1390, table=0, priority=40000, selector=[ETH_TYPE:bddp], treatment=[immediate=[OUTPUT:CONTROLLER], clearDeferred]
    ADDED, bytes=462, packets=11, table=0, priority=40000, selector=[ETH_TYPE:arp], treatment=[immediate=[OUTPUT:CONTROLLER], clearDeferred]
    ADDED, bytes=108420, packets=1390, table=0, priority=40000, selector=[ETH_TYPE:lldp], treatment=[immediate=[OUTPUT:CONTROLLER], clearDeferred]
    ADDED, bytes=0, packets=0, table=0, priority=5, selector=[ETH_TYPE:arp], treatment=[immediate=[OUTPUT:CONTROLLER], clearDeferred]
    ADDED, bytes=196, packets=2, table=0, priority=5, selector=[ETH_TYPE:ipv4], treatment=[immediate=[OUTPUT:CONTROLLER], clearDeferred]
deviceId=of:000000000000000d, flowRuleCount=5
    ADDED, bytes=420, packets=10, table=0, priority=40000, selector=[ETH_TYPE:arp], treatment=[immediate=[OUTPUT:CONTROLLER], clearDeferred]
    ADDED, bytes=108420, packets=1390, table=0, priority=40000, selector=[ETH_TYPE:bddp], treatment=[immediate=[OUTPUT:CONTROLLER], clearDeferred]
    ADDED, bytes=108576, packets=1392, table=0, priority=40000, selector=[ETH_TYPE:lldp], treatment=[immediate=[OUTPUT:CONTROLLER], clearDeferred]
    ADDED, bytes=0, packets=0, table=0, priority=5, selector=[ETH_TYPE:arp], treatment=[immediate=[OUTPUT:CONTROLLER], clearDeferred]
    ADDED, bytes=0, packets=0, table=0, priority=5, selector=[ETH_TYPE:ipv4], treatment=[immediate=[OUTPUT:CONTROLLER], clearDeferred]
deviceId=of:000000000000000e, flowRuleCount=5
    ADDED, bytes=108420, packets=1390, table=0, priority=40000, selector=[ETH_TYPE:lldp], treatment=[immediate=[OUTPUT:CONTROLLER], clearDeferred]
    ADDED, bytes=462, packets=11, table=0, priority=40000, selector=[ETH_TYPE:arp], treatment=[immediate=[OUTPUT:CONTROLLER], clearDeferred]
    ADDED, bytes=108420, packets=1390, table=0, priority=40000, selector=[ETH_TYPE:bddp], treatment=[immediate=[OUTPUT:CONTROLLER], clearDeferred]
    ADDED, bytes=0, packets=0, table=0, priority=5, selector=[ETH_TYPE:arp], treatment=[immediate=[OUTPUT:CONTROLLER], clearDeferred]
    ADDED, bytes=196, packets=2, table=0, priority=5, selector=[ETH_TYPE:ipv4], treatment=[immediate=[OUTPUT:CONTROLLER], clearDeferred]
```

Flows for Device of:0000000000000001 (5 Total)

Search | Search By

STATE	PACKETS	DURATION	FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT	APP NAME
Added	0	1,802	5	0	ETH_TYPE:arp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	4	1,802	5	0	ETH_TYPE:ipv4	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	2,528	1,959	40000	0	ETH_TYPE:bddp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	2,531	1,959	40000	0	ETH_TYPE:lldp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	0	1,959	40000	0	ETH_TYPE:arp	imm[OUTPUT:CONTROLLER], cleared:true	*core

Ilustración 52 - Análisis de flujos de datos

En la ilustración se puede observar cómo ONOS proporciona muchos detalles sobre los flujos en los switches. Por ejemplo, cada entrada de flujo define un selector y un tratamiento que es el conjunto de tráfico que coincide con la entrada de flujo y cómo se debe manejar este tráfico. Se puede observar también que cada entrada de flujo se etiqueta con un ID de aplicación, este ID de aplicación identifica qué aplicación instaló esta entrada de flujo. Esta es una característica útil que puede ayudar a un administrador a identificar qué aplicación puede estar funcionando mal o consumiendo muchos recursos.
ONOS nos permite calcular las rutas más cortas entre dos nodos. Esto es especialmente útil para la convergencia de la red.

Por ejemplo, si queremos conocer el coste entre el Spine-1 y el Leaf-4 basta con ejecutar el comando ***path y TAB*** automáticamente saldrán la lista de equipos con sus identificadores, en este caso 00000000000001 es Spine-1 y 0000000000000e es Leaf-4, entre ambos existe un coste de 1.

```
onos> paths of:0000000000000001 of:000000000000000e
of:0000000000000001/4-of:000000000000000e/1; cost=1.0
```

Ilustración 53 - Coste de ruta entre Spine-1 y Leaf-4

Realizando el mismo proceso para el Spine1 y el Spine2, el coste de ruta entre ellos es de 2.

```
onos> paths of:0000000000000001 of:0000000000000002
of:0000000000000001/3-of:000000000000000d/1==>of:000000000000000d/2-of:0000000000000002/3; cost=2.0
of:0000000000000001/2-of:000000000000000c/1==>of:000000000000000c/2-of:0000000000000002/2; cost=2.0
of:0000000000000001/1-of:000000000000000b/1==>of:000000000000000b/2-of:0000000000000002/1; cost=2.0
of:0000000000000001/4-of:000000000000000e/1==>of:000000000000000e/2-of:0000000000000002/4; cost=2.0
```

Ilustración 54 - Coste de ruta entre Spine-1 y Spine-2

Si recordamos lo analizado en la arquitectura Clos el coste de ruta entre hots está siempre a 3 saltos. Y como se puede observar, esta regla se cumple.

Con la ayuda de la herramienta Wireshark podemos analizar una captura de tráfico entre el controlador ONOS y cualquiera de los switches que utilizan OpenFlow v1.3. Después de activar la aplicación OpenFlow, se deberían de ver una serie de mensajes OpenFlow en Wireshark como se muestra en la siguiente ilustración. Para una mejor visualización de la captura, se puede aplicar el filtro ***"Openflow_v4"***

```
51404 24.864151320  172.17.0.6   172.17.0.1   OpenFlow   90 Type: OFPT_ROLE_REQUEST
51406 24.864217780  172.17.0.6   172.17.0.1   OpenFlow   90 Type: OFPT_ROLE_REQUEST
51408 24.864240115  172.17.0.1   172.17.0.6   OpenFlow   74 Type: OFPT_BARRIER_REPLY
51409 24.864270695  172.17.0.1   172.17.0.5   OpenFlow  186 Type: OFPT_PACKET_IN
51410 24.864271280  172.17.0.6   172.17.0.1   OpenFlow   90 Type: OFPT_ROLE_REQUEST
51413 24.864520165  172.17.0.1   172.17.0.6   OpenFlow  186 Type: OFPT_PACKET_IN
51414 24.864530385  172.17.0.1   172.17.0.5   OpenFlow  186 Type: OFPT_PACKET_IN
51416 24.864636095  172.17.0.1   172.17.0.6   OpenFlow  186 Type: OFPT_PACKET_IN
```

Ilustración 55 - Captura Openflow

El primer mensaje que vemos es un ***OFPT_FEATURES_REQUEST*** del controlador 172.17.0.6 (ONOS_2) al switch 172.17.0.1. Entre otros atributos de capacidad, este mensaje se utiliza para encontrar el identificador de ruta de datos (DPID) del switch. El DPID identifica de forma única una ruta de datos en la topología de OpenFlow y se crea de forma dinámica al combinar la dirección MAC del dispositivo en los 48 bits inferiores, precedida de una cadena de 16 bits que determinará el implementador.

```
> Frame 51393: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface docker0, id 0
> Ethernet II, Src: 02:42:ac:11:00:06 (02:42:ac:11:00:06), Dst: 02:42:69:e9:1d:b9 (02:42:69:e9:1d:b9)
> Internet Protocol Version 4, Src: 172.17.0.6, Dst: 172.17.0.1
> Transmission Control Protocol, Src Port: 6653, Dst Port: 37868, Seq: 773005, Ack: 1243561, Len: 24
v OpenFlow 1.3
    Version: 1.3 (0x04)
    Type: OFPT_ROLE_REQUEST (24)
    Length: 24
    Transaction ID: 30541
    Role: OFPCR_ROLE_MASTER (0x00000002)
    Pad: 00000000
    Generation ID: 0x0000000000000000
```

Ilustración 56 – Mensaje OpenFlow Request

El switch responde con un mensaje ***OFPT_FEATURES_REPLY***.

```
> Frame 51518: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface docker0, id 0
> Ethernet II, Src: 02:42:69:e9:1d:b9 (02:42:69:e9:1d:b9), Dst: 02:42:ac:11:00:06 (02:42:ac:11:00:06)
> Internet Protocol Version 4, Src: 172.17.0.1, Dst: 172.17.0.6
> Transmission Control Protocol, Src Port: 37868, Dst Port: 6653, Seq: 1245265, Ack: 773629, Len: 24
v OpenFlow 1.3
    Version: 1.3 (0x04)
    Type: OFPT_ROLE_REPLY (25)
    Length: 24
    Transaction ID: 30566
    Role: OFPCR_ROLE_MASTER (0x00000002)
    Pad: 00000000
    Generation ID: 0x0000000000000000
```

Ilustración 57 – Mensaje OpenFlow Reply

Cuando se inicia la operación de OpenFlow, el switch debe conectarse a todos los controladores con los que está configurado e intentar mantener la conectividad con todos ellos al mismo tiempo. Muchos controladores pueden enviar comandos de controlador a switch. Un controlador puede solicitar que se cambie su rol a

OFPCR_ROLE_MASTER. Esta función es similar a ***OFPCR_ROLE_EQUAL*** y tiene acceso total al conmutador, la diferencia es que el conmutador garantiza que es el único controlador en esta función.

El controlador puede solicitar el estado de la ruta de datos mediante el mensaje ***OFPT_MULTIPART_REQUEST***. Los tipos de mensajes manejados por este mensaje incluyen varias estadísticas.

En el ejemplo, ***OFPT_PORT_STATS*** el controlador obtiene estadísticas de tráfico de sus puertos. En el ejemplo, se pueden ver las estadísticas del puerto 4.

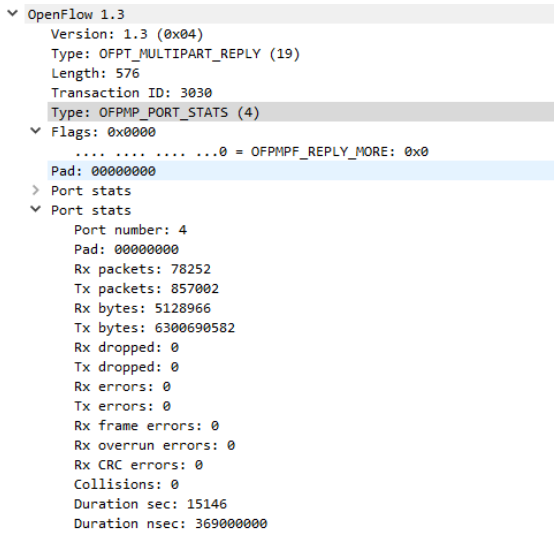


Ilustración 58 – Mensaje OpenFlow Stats

El controlador también solicita las estadísticas de flujo mediante el envío de una solicitud de varias partes ***OFPMP_FLOW***. Los switches responden con un mensaje ***OFPMP_FLOW_MULTIPART_REPLY*** que contiene la información solicitada:

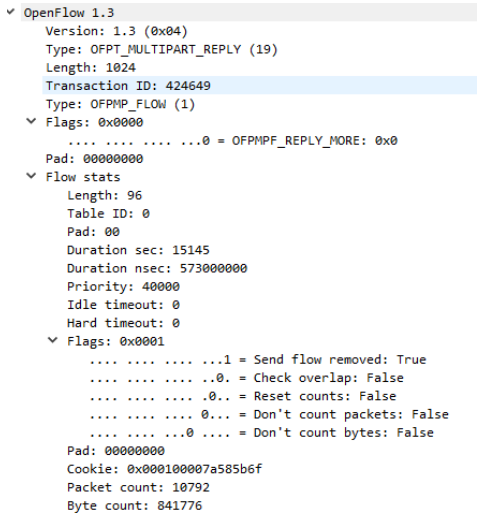


Ilustración 59 - Mensaje OpenFlow Multipart

Cuando se genera tráfico el controlador comienza a hacer su trabajo y responde a los mensajes ***PACKET_IN*** con paquetes ***PACKET_OUT*** que incluyen una lista de acciones. Esta lista de acciones generalmente contendrá una acción de salida, que especificará un número de puerto. Cuando el controlador tiene un paquete de datos para reenviar a través del switch, utiliza el mensaje OpenFlow ***PACKET_OUT***, dicho paquete de datos proveniente del controlador puede tomar dos rutas diferentes a través de la lógica OpenFlow, ya sea que especifique directamente el puerto de salida y el paquete sea pasa a ese puerto, o el controlador indica que desea diferir la decisión de reenvío a la lógica de coincidencia de paquetes. En este caso, el controlador dicta esto estipulando el puerto virtual ***TABLE*** como el puerto de salida.

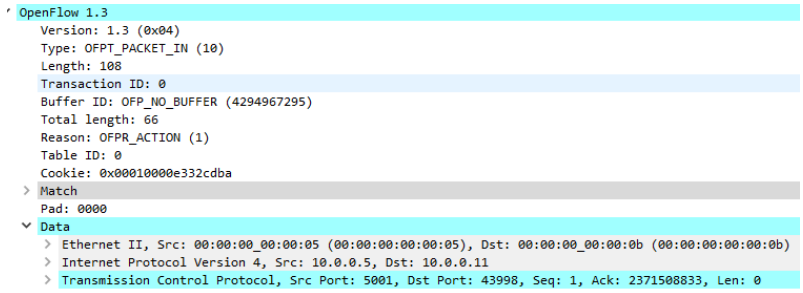


Ilustración 60 - Mensaje OpenFlow Packet_In

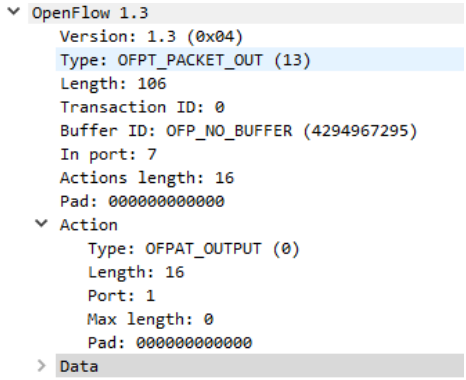


Ilustración 61 – Mensaje OpenFlow Packet_out

En este ejemplo de captura, se puede observar como el controlador especifica el puerto de salida 1 desde el puerto de entrada 7.

Además de esto, el controlador también envía paquetes ***FLOW_MOD*** como reacción a los mensajes ***PACKET_IN*** para realizar una modificación de la tabla de flujo.

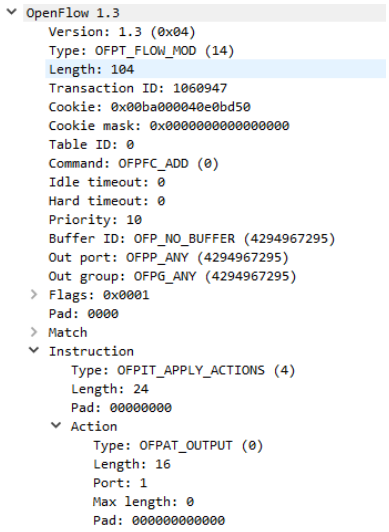
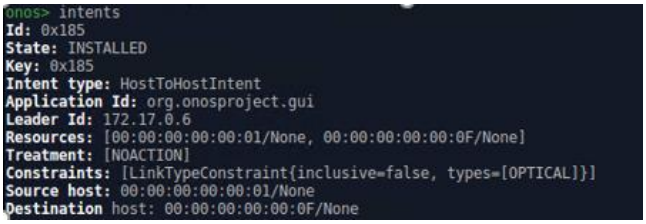


Ilustración 62 – Mensaje OpenFlow Flow_Mod

Otra característica que podemos realizar con ONOS es seleccionar una ruta por defecto o programar la comunicación de uno o varios equipos a través de una ruta predefinida. Esto es muy útil cuando necesitamos que el tráfico pase por unos equipos específicos o en caso de caída de un enlace, obligar a que el tráfico utilice automáticamente el encaminamiento definido. En ONOS esto se puede realizar a través del comando ***add-host-intent.*** Si pulsamos la tecla ***TAB*** automáticamente saldrán una lista de equipos para elegir.

Ilustración 63 - Creación de Intent



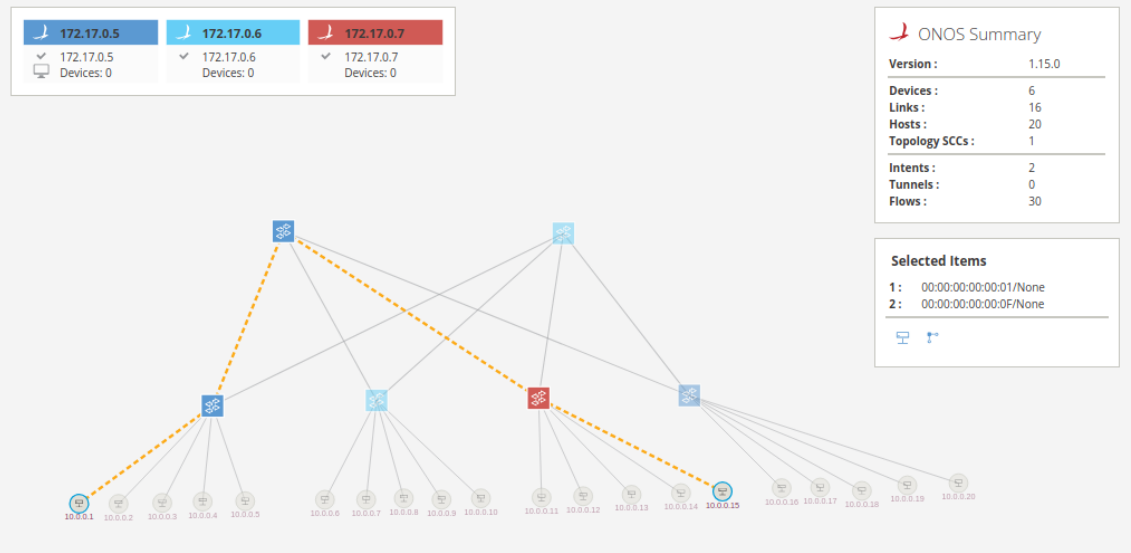


Ilustración 64 - Visualización de la ruta automatizada creada

El resultado es el mostrado, en el podemos ver como los hots seleccionados establecen una ruta preprogramada en caso de fallo, elegiendo el Spine1 en caso de fallo del Spine2 como ruta alternativa más corta.

Otro escenario que nos podemos encontrar en cualquier situación dentro de las redes es la caída del equipo central o Core, cuando un nodo de Core queda incomunicado, se había analizado que la viabilidad de la infraestructura aun utilizando mecanismos de balanceo de cargas, se veía afectada por la pérdida de los enlaces, que dependiendo de la topología y la tecnología configurada se situaba sobre un 25% o incluso un 50%. Esta situación no es óptima para CPDs de gran escala. Por lo que a continuación, se expondrá un ejemplo de balanceo entre controladores SDN y como su uso ayuda a este tipo de problemas.

Partimos de un escenario donde los 3 nodos en clúster está previamente configurados en modo Master y Esclavo.

```
onos> devices -s
id=of:0000000000000001, available=true, role=MASTER, type=SWITCH, driver=ovs
id=of:0000000000000002, available=true, role=STANDBY, type=SWITCH, driver=ovs
id=of:000000000000000b, available=true, role=MASTER, type=SWITCH, driver=ovs
id=of:000000000000000c, available=true, role=STANDBY, type=SWITCH, driver=ovs
id=of:000000000000000d, available=true, role=STANDBY, type=SWITCH, driver=ovs
id=of:000000000000000e, available=true, role=STANDBY, type=SWITCH, driver=ovs
```

Ilustración 65 – Estado Maestro/Esclavo del Clúster

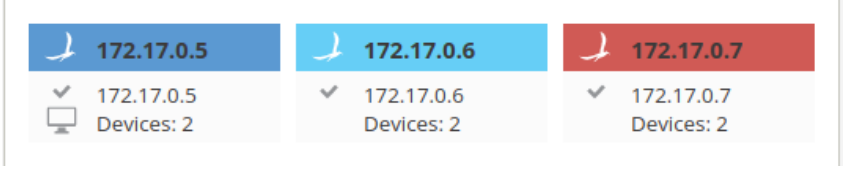


Ilustración 66 - Estado Inicial del Clúster

Ahora activamos el modo de balanceo de carga activando la App ***org.onosproject.mlb*** y tirando la conexión de ONOS_3 con IP. 172.17.0.7, el resultado es el siguiente. Se puede apreciar como el controlador ONOS_1 con la IP.172.17.0.5 está activando sus

enlaces secundarios y asume el tráfico del controlador ONOS_3 que ha perdido sus enlaces.

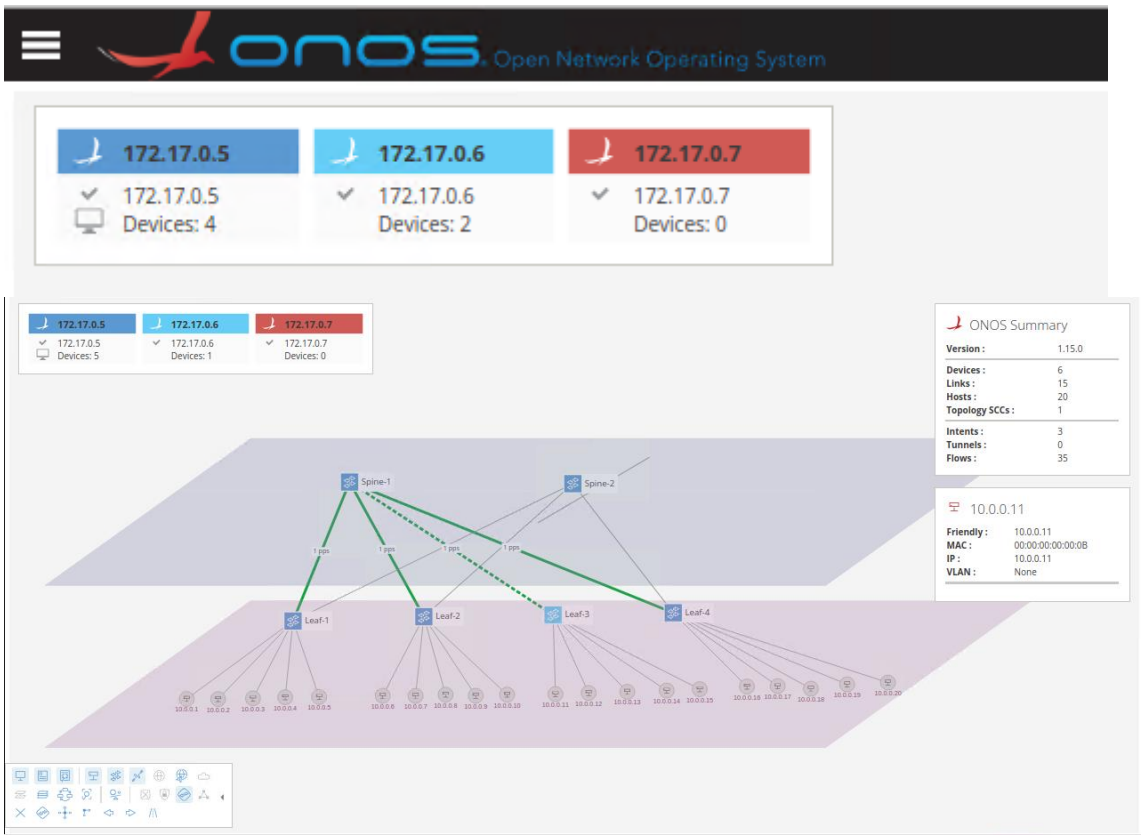


Ilustración 67 – Balanceo activado con ejemplo de fallo en controlador ONOS_3

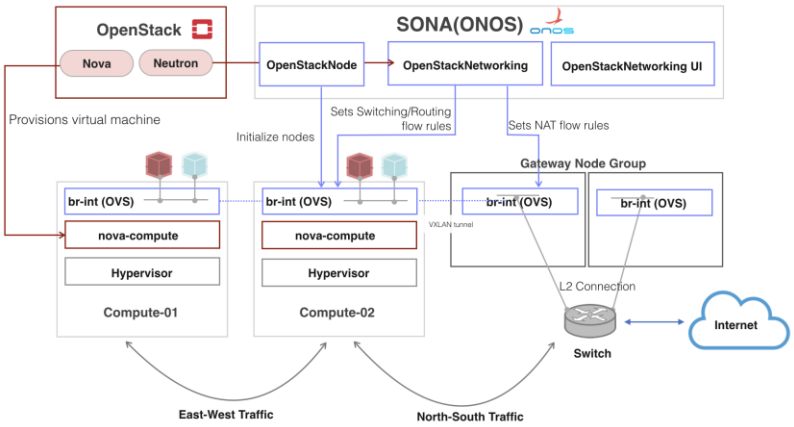
Podemos apreciar la topología Overlay donde el balanceo de cargas y el uso de tráfico por los enlaces se hace efectivo. Sin suponer ninguna interrupción en la infraestructura. Tanto los switches Spine-Leaf como los hosts no han notado corte de comunicación.

8. LAB 2: Despliegue de tuneles overlay en OpenvSwitch (OVS)

8.1 Elección de escenario a desplegar

La idea principal era ampliar el laboratorio anteriormente desplegado con la instalación de una arquitectura SONA o Arquitectura de red de superposición simplificada, que es un servicio de virtualización de red de inquilinos optimizado para centros de datos

basados en la nube. De este modo tendríamos tres aplicaciones: ONOS, SONA y Openstack.



- **Red de gestión:** Se utiliza para que ONOS controle los switches virtuales y OpenStack para comunicarse con el agente de cómputo nova que se ejecuta en el nodo de cómputo.

- **Red de datos:** Se utiliza para el tráfico Este-Oeste a través del túnel VXLAN, GRE o GENEVE
- **Red externa:** Se utiliza para el tráfico Norte-Sur, normalmente solo los nodos de puerta de enlace tienen acceso a esta red

Sin embargo, aun realizando los pasos descritos por Onos Project en la url que se especifica a continuación:
https://wiki.onosproject.org/display/ONOS/All-in-one+Installation+Guide

No se ha podido desplegar el escenario completo y ante la falta de actualización por parte del personal técnico que soporta SONA dentro de la comunidad ONF,se ha descartado su implementación para la realización de este trabajo.

Finalmente, el escenario a desplegar será la creación de maqueta compuesta por dos máquinas virtuales para estudiar y analizar el funcionamiento de las tecnologías VXLAN, GENEVE y LISP a través de las opciones que nos brinda OpenVSwitch (OVS).

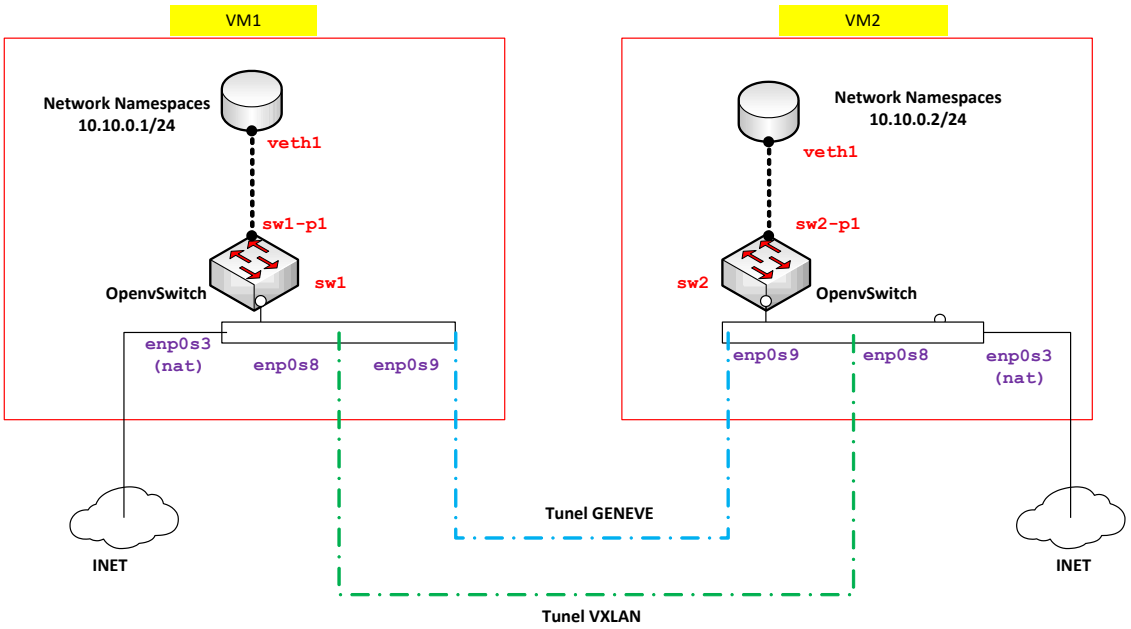


Ilustración 68 – Escenario virtual con despliegue de túneles overlay (elaboración propia)

8.2 Análisis e implementación

OpenVSwitch es software open source que nos permite emular un switch virtual multicapa diseñado para comunicar instancias virtuales en entornos de virtualización. Con OVS se puede reenviar tráfico entre diferentes máquinas virtuales en el mismo equipo físico o comunicar estas máquinas a través de una red física. Entre sus funcionalidades OpenVSwitch soporta VLAN, Netflow, sFlow, OpenFlow, STP, QoS y permite desplegar túneles de red como GRE, STT, VXLAN, GENEVE o LISP.

En la Ilustración 69, se muestra un esquema lógico del escenario desplegado mediante el uso de dos máquinas virtuales con su correspondiente switch OVS configurado a nivel de capa 2. La configuración de Network Namespaces ***"10.10.0.1/24"*** y ***"10.10.0.2/24"*** nos proporcionarán instancias independientes de interfaces de red y tablas de enrutamiento para conseguir el aislamiento del funcionamiento de la red. Los Network Namespaces se interconectan con el puerto ethernet de OpenVSwitch a través de ethernet virtuales como ***"veth1".*** Los puertos ethernet virtuales son equivalentes a un par de interfaces ethernet físicos interconectadas por un cable, pero en este caso, implementadas únicamente mediante software.

El despliegue se realiza mediante la creación de dos máquinas virtuales ***VM1*** y ***VM2*** desplegadas dentro de Virtual Box. Se ha definido previamente las siguientes interfaces de red (Host-only) para crear un escenario virtual.

```
- vboxnet0 192.168.70.1/24
- vboxnet1 192.168.80.1/24
```

```
bcastrose@VM1:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::6fb9:ba80:a79c:60e5  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:3c:a7:8c  txqueuelen 1000  (Ethernet)
        RX packets 136439  bytes 197659101 (197.6 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 20464  bytes 1732977 (1.7 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.70.11  netmask 255.255.255.0  broadcast 192.168.70.255
        inet6 fe80::487c:b219:33b9:d319  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:4d:55:df  txqueuelen 1000  (Ethernet)
        RX packets 25  bytes 14930 (14.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 68  bytes 9069 (9.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s9: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.80.11  netmask 255.255.255.0  broadcast 192.168.80.255
        inet6 fe80::a0af:911e:4648:ea11  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:a3:27:3d  txqueuelen 1000  (Ethernet)
        RX packets 17  bytes 5790 (5.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 67  bytes 8962 (8.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Ilustración 69 - Interfaces de red VM1

```
bcastrose@VM2:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::80eb:9416:4bad:c82  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:28:1d:00  txqueuelen 1000  (Ethernet)
        RX packets 41  bytes 28924 (28.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 77  bytes 12650 (12.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.70.13  netmask 255.255.255.0  broadcast 192.168.70.255
        inet6 fe80::7c38:97cb:e87c:ca48  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:03:c3:3d  txqueuelen 1000  (Ethernet)
        RX packets 6  bytes 5218 (5.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 39  bytes 4777 (4.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s9: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.80.12  netmask 255.255.255.0  broadcast 192.168.80.255
        inet6 fe80::f73d:89c4:908a:74fc  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:56:44:5c  txqueuelen 1000  (Ethernet)
        RX packets 1  bytes 590 (590.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 40  bytes 4854 (4.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Ilustración 70 - Interfaces de red VM2

El sistema operativo instalado en cada máquina es Kubuntu 22.04 LTS pero cualquier otro sistema operativo con kernel Linux es válido. Una vez configuradas las interfaces de red de la VM1 y la VM2, es necesario optimizar el sistema operativo y actualizar el kernel a la versión más actual, que en el momento de escribir este apartado es la v.6.0.1. Los pasos para actualizar el kernel se pueden consultar en el Anexo 12.3.1.

```
bcastrose@VM2:~$ uname -r
6.0.1-060001-generic
```

Ilustración 71 - V.6.0.1 Kernel

El siguiente paso es instalar OpenVSwitch para poder desplegar los servicios necesarios para la creación de túneles superpuestos. Si utilizamos Kubuntu 22.04 ***"openvswitch-switch"*** y ***"openvswitch-common"*** se incluyen dentro de los componentes principales. También están disponibles paquetes adicionales para soporte de documentación, ipsec, pki, VTEP y Python. La ruta de datos del kernel de OpenVSwitch se mantiene como parte del kernel disponible en la distribución. Por lo que solo es necesario ejecutar el siguiente comando para instalarlo.

```
sudo apt-get install openvswitch-switch-dpdk
```

```
bcastrose@VM1:~$ sudo ovs-vsctl show
0bd0bae5-7a57-4837-b87d-fbfdd71cb08b
   ovs_version: "3.0.1
bcastrose@VM2:~$ sudo ovs-vsctl show
1eed137b-7793-4cb1-b287-6ea270306ddb
    ovs_version: "3.0.1"
```

Una vez instalados los requisitos previos y desplegado OVS, realizamos la configuración de las interfaces para cada máquina virtual. Empezando por la definición de los nombres de red ***"vm1"*** y ***"vm2"*** para cada máquina. El siguiente comando es la definición de los enlaces ***"veth1"*** que presentan dos extremos, normalmente uno hace referencia a la máquina y otro a los network namespace. Todos los paquetes entran por un extremo salen automáticamente por el otro. Por último, se crean los switches con el comando ***ovs-vsctl add-br*** para ***"sw1"*** y ***"sw2"*** y finalizamos con la definición de los puertos a desplegar ejecutando los siguientes comandos: ***ovs-vsctl add-port*** "***sw1-p1"*** y "***sw2-p1".***

- **VM1:**

```
sudo ip netns add vm1
sudo ip link add name veth1 type veth peer name sw1-p1
sudo ip link set dev veth1 netns vm1
sudo ip netns exec vm1 ifconfig veth1 10.10.0.1/24 up
sudo ovs-vsctl add-br sw1
sudo ovs-vsctl add-port sw1 sw1-p1
sudo ip link set sw1-p1 up
sudo ip link set sw1 up
```

- **VM2:**

```
sudo ip netns add vm2
sudo ip link add name veth1 type veth peer name sw2-p1
sudo ip link set dev veth1 netns vm2
sudo ip netns exec vm2 ifconfig veth1 10.10.0.2/24 up
sudo ovs-vsctl add-br sw2
sudo ovs-vsctl add-port sw2 sw2-p1
sudo ip link set sw2-p1 up
sudo ip link set sw2 up
```

Las dos máquinas ya se comunican entre sí y ambas tienen desplegados ambos switches OVS con sus interfaces definidas.

A continuación, se crea un túnel definido como ***"tun0"*** que permitirá utilizar la tecnología VXLAN para comunicar las interfaces enp0s8 de cada máquina.

- **VM1:**

```
sudo ovs-vsctl add-port sw1 tun0 -- set interface tun0 type=vxlan
options:remote_ip=192.168.70.13 options:key=5000
```

- **VM2:**

```
sudo ovs-vsctl add-port sw2 tun0 -- set interface tun0 type=vxlan
options:remote_ip=192.168.70.11 options:key=5000
```

El resultado, es la creación del túnel "vxlan_sys_4789" en cada máquina. El número 4789 corresponde al puerto origen/destino de VXLAN.

```
vxlan_sys_4789: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 65000
        inet6 fe80::1869:6cff:fed4:924a  prefixlen 64  scopeid 0x20<link>
        ether c6:1f:1d:fa:2b:f4  txqueuelen 1000  (Ethernet)
        RX packets 185  bytes 14923 (14.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 191  bytes 15333 (15.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Ilustración 72 - Túnel VXLAN

Para comprobar la estabilidad del túnel VXLAN se genera tráfico ICMP desde la VM1 con IP. 10.0.0.1 hacia la VM2 con IP.10.0.0.2. La captura de tráfico realizada través de Wireshark se puede ver en la Ilustración 73.

No.	Time	Source	Destination	Protocol	Length	Info
17	7.167656989	10.10.0.1	10.10.0.2	ICMP	98	Echo (ping) request id=0xfdc8, seq=725/54530, ttl=64 (reply in 18)
18	7.167939806	10.10.0.2	10.10.0.1	ICMP	98	Echo (ping) reply id=0xfdc8, seq=725/54530, ttl=64 (request in 17)
19	8.193740799	10.10.0.1	10.10.0.2	ICMP	98	Echo (ping) request id=0xfdc8, seq=726/54786, ttl=64 (reply in 20)
20	8.194085712	10.10.0.2	10.10.0.1	ICMP	98	Echo (ping) reply id=0xfdc8, seq=726/54786, ttl=64 (request in 19)
21	9.217168079	10.10.0.1	10.10.0.2	ICMP	98	Echo (ping) request id=0xfdc8, seq=727/55042, ttl=64 (reply in 22)
22	9.217496736	10.10.0.2	10.10.0.1	ICMP	98	Echo (ping) reply id=0xfdc8, seq=727/55042, ttl=64 (request in 21)
23	9.343726620	86:5f:1d:af:a3:af	1a:c5:b9:83:7c:d6	ARP	42	Who has 10.10.0.2? Tell 10.10.0.1
24	9.343998869	1a:c5:b9:83:7c:d6	86:5f:1d:af:a3:af	ARP	42	10.10.0.2 is at 1a:c5:b9:83:7c:d6
25	10.240237044	10.10.0.1	10.10.0.2	ICMP	98	Echo (ping) request id=0xfdc8, seq=728/55298, ttl=64 (reply in 26)
26	10.240488589	10.10.0.2	10.10.0.1	ICMP	98	Echo (ping) reply id=0xfdc8, seq=728/55298, ttl=64 (request in 25)

```
> Frame 17: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface vxlan_sys_4789, id 0
> Ethernet II, Src: 86:5f:1d:af:a3:af (86:5f:1d:af:a3:af), Dst: 1a:c5:b9:83:7c:d6 (1a:c5:b9:83:7c:d6)
> Internet Protocol Version 4, Src: 10.10.0.1, Dst: 10.10.0.2
> Internet Control Message Protocol
```

Ilustración 73 - Captura wireshark VXLAN

Si analizamos el frame, VXLAN presenta 148 bytes que consiste en un VNI de 24 bits (campo VNID) y algunos bits reservados. El encabezado de VXLAN junto con la trama de Ethernet original se incluye en la carga útil de UDP.

VxLAN Ethernet 802.1Q frame - 1596 bytes/octets															
Pre SFD	O-DMAC	O-SMAC	O-TAG 0x0800	Type 0x8100	Outer IP header	O-UDP Header	VxLAN Header	I-DMAC	I-SMAC	C-TAG 0x8100	Type 0x0800	IP 20 bytes	TCP 20 bytes	Payload / Data 6-1460	CRC FCS	IFG
8	6	6	4	2	20	8	8	6	6	4	2	46-1500			4	12

Ilustración 74 - Frame VXLAN

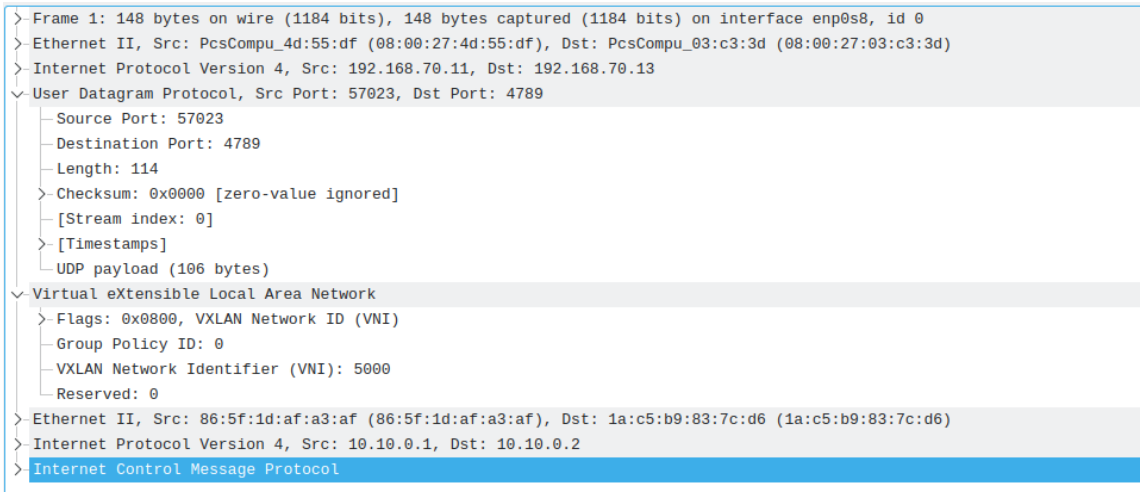


Ilustración 75 - Captura Frame 1

En la Ilustración 74, se puede observar la creación del túnel Vxlan y su VNI identificativo (5000), así como el tráfico de red de superposición de la VM1. Literalmente no hay nada sustancial que analizar porque se trata de un tráfico ICMP clásico, donde primero se envía un paquete y se responde con una solicitud ARP de difusión, para volver a enviar otra solicitud ICMP.Es decir, tráfico de envío y respuesta. No obstante, aunque VM1 y VM2 están físicamente en diferentes redes de capa 2, creen estar en la misma red gracias a la tunelización de VXLAN en este caso.

Analizando los paquetes de la red subyacente, cuando el frame de capa 2 con la dirección de destino (08:00:27:03:c3:3d) llega al puerto al ser parte del tráfico BUM, necesita ser enviado a la interfaz física con IP. 10.10.0.1. Sin embargo, el puerto también necesita enviar una solicitud ARP para conocer la dirección MAC de la VM2. Por lo tanto, el primer paquete en la red subyacente es la solicitud ARP que consulta la dirección MAC de la máquina VM1 con IP.10.10.0.1.

El tercer paquete es la solicitud ARP de la red superpuesta. Wireshark muestra dos tramas Ethernet con direcciones MAC de origen/destino debido a que una corresponde a la red de la capa subyacente y la otra es la red de la capa superpuesta.

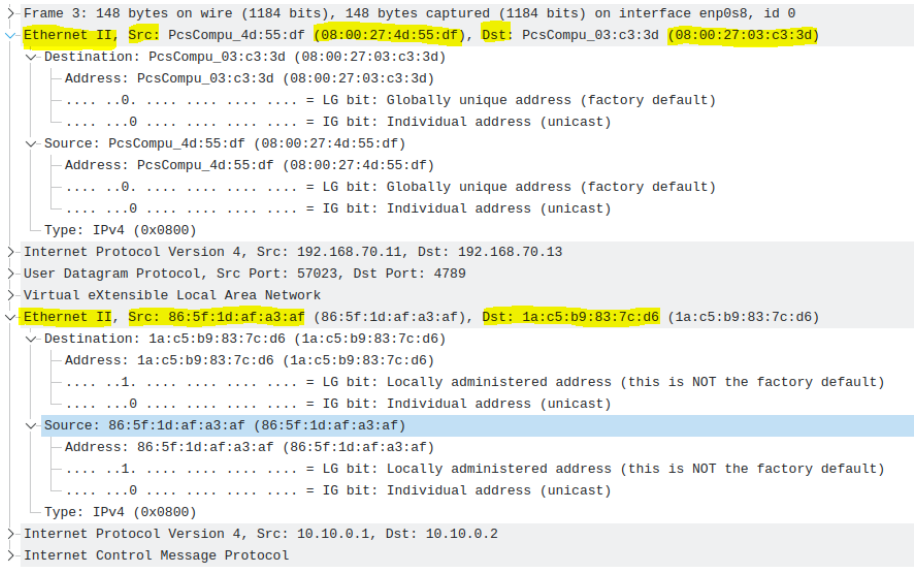


Ilustración 76 - Captura Frame 3

OpenVSwitch actualmente no admite compatibilidad con los aspectos de multidifusión de VXLAN. Para sortear la falta de compatibilidad con multidifusión, es posible provisionar previamente las asignaciones de direcciones MAC e IP de forma manual o a través de un controlador.
De manera predeterminada, OpenVSwitch utilizará el puerto IANA asignado para VXLAN, que es 4789. Sin embargo, es posible configurar el puerto UDP de destino manualmente en función del túnel VXLAN que se desee.

Node A	Node B	Total Frames	Total Data	Frames A → B	Data A → B	Frames B → A	Data B → A	Relative Start	Total Duration	Rate A → B	Rate B → A
192.168.70.11	192.168.70.13	100	14.2 KB	50	7.1 KB	50	7.1 KB	0	48.19209	1.2 Kbits/s	1.2 Kbits/s
10.10.0.1	10.10.0.2	96	13.9 KB	48	6.9 KB	48	6.9 KB	0	48.19209	1.2 Kbits/s	1.2 Kbits/s

Ilustración 77 - Estadísticas del tráfico generado VXLAN

Para el despliegue de un túnel GENEVE, el proceso es muy similar al despliegue del túnel VXLAN, sin embargo, la configuración se va a realizar entre ambas interfaces enps09 de cada máquina.

- **VM1:**

```
sudo ovs-vsctl add-port sw1 tun0 -- set interface tun0 type=geneve
options:remote_ip=192.168.80.12 options:key=7000
```

- **VM2:**

```
sudo ovs-vsctl add-port sw2 tun0 -- set interface tun0 type=geneve
options:remote_ip=192.168.80.11 options:key=7000
```

El resultado, es la creación del túnel “geneve_sys_6081” en cada máquina. El número 6081 corresponde al puerto destino de GENEVE.

```
genev_sys_6081: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 65000
        inet6 fe80::6c25:edff:fe1f:8933  prefixlen 64  scopeid 0x20<link>
        ether 86:53:57:e9:81:4b  txqueuelen 1000  (Ethernet)
        RX packets 23  bytes 1820 (1.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 23  bytes 1820 (1.8 KB)
        TX errors 0  dropped 21 overruns 0  carrier 0  collisions 0
```

Ilustración 78 - Tunel GENEVE

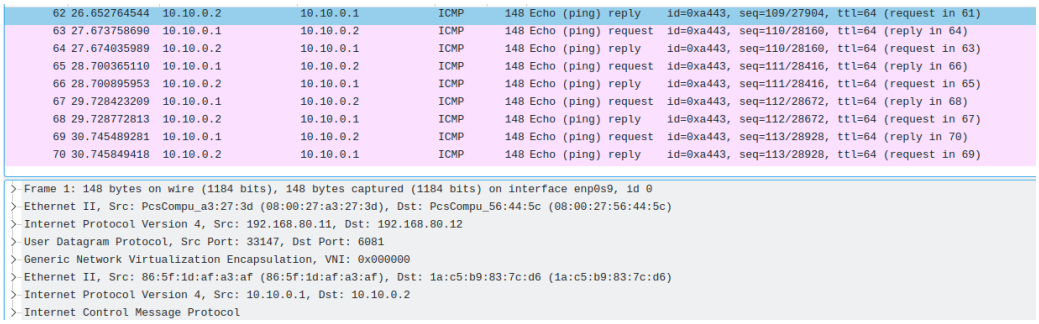


Ilustración 79 - Captura Wireshark GENEVE

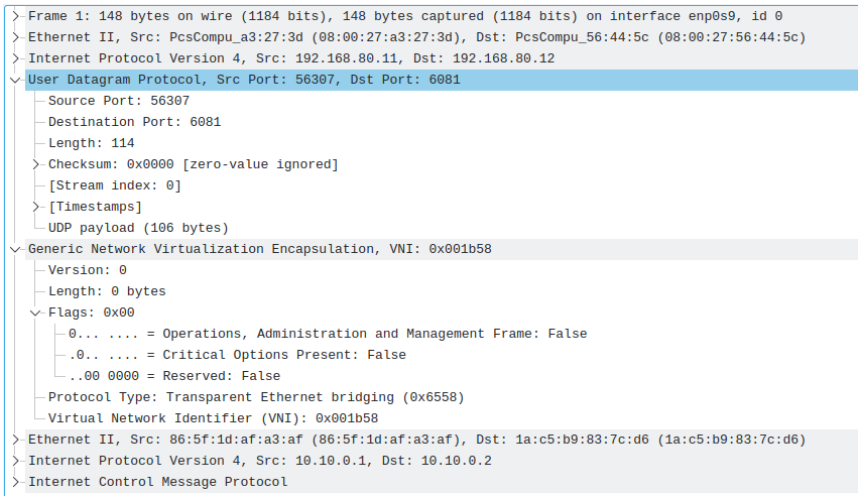


Ilustración 80 - Captura Frame 1

El segmento UDP asocia el puerto de destino UDP 6081.IANA ha asignado puerto 6081 como el puerto de destino fijo para GENEVE.Wireshark decodificar el encabezado Geneve con un (VNI) de 0x001b58, además muestra dos tramas Ethernet con direcciones MAC de origen/destino debido a que una corresponde a la red de la capa subyacente y la otra es la red de la capa superpuesta. Igual que ocurría con el túnel VXLAN.

Node A	Node B	Total Frames	Total Data	Frames A → B	Data A → B	Frames B → A	Data B → A	Relative Start	Total Duration	Rate A → B	Rate B → A
192.168.80.11	192.168.80.12	76	10.7 KB	39	5.5 KB	37	5.2 KB	0	34.91253	1.3 Kbits/s	1.2 Kbits/s
10.10.0.1	10.10.0.2	70	10.1 KB	35	5.1 KB	35	5.1 KB	0	34.91253	1.2 Kbits/s	1.2 Kbits/s

Ilustración 81 - Estadísticas de tráfico generado GENEVE

Por último, se pretendía desplegar un túnel LISP en OpenVSwitch, pero debido a que LISP utiliza un mecanismo de tunelizado de capa 3, los paquetes encapsulados no llevarían encabezados de Ethernet y las solicitudes ARP no se enviarían a través del túnel. Debido a esto y hasta que se mejore la compatibilidad con los túneles de capa 3 por parte de OpenVSwitch se ha descartado su configuración y posterior análisis con Wireshark en este apartado.

9. Conclusiones

En el presente Trabajo de Fin de Grado se han desplegado dos laboratorios propios para el estudio de las redes SDN mediante el uso de programas y dispositivos Open Source como Mininet, ONOS, OpenFlow y OpenVSwitch.

Como se ha podido ver, las redes definidas por software no son un paradigma nuevo pero su implementación por parte de las empresas hoy en día sigue estando muy por debajo de las cualidades que esta tecnología puede ofrecer. SDN nos ayuda a solventar algunos de los problemas que presentan las redes de datos tradicionales en cuanto a flexibilidad, automatización y facilidad de gestión.

El estudio realizado en este trabajo sea centrado en primer lugar, en la implementación de la tecnología SDN dentro de una arquitectura Clos. Su intuitiva instalación, así como la configuración de algunos de los módulos que componen el plano de control y las APIS integradas en el propio controlador, abren un sinfín de posibilidades para gestionar una red basada en software.

Además, la instalación en conjunto con las herramientas Mininet y Miniedit ha proporcionado la base para un despliegue limpio de un escenario SDN. La elección de instalación de un clúster de balanceadores bajo el controlador ONOS, nos ha permitido estudiar el funcionamiento del protocolo OpenFlow. La gestión automatizada de flujos de tráfico y el balanceo de rutas entre los elementos de la topología desplegada, ha verificado la viabilidad de la solución bajo el dominio de un controlador SDN.

Siguiendo esta dinámica, para complementar el estudio de las redes SDN, se han desplegado dos máquinas virtuales que con la ayuda de OpenVSwitch, nos ha permitido estudiar el comportamiento de las tecnologías VXLAN y GENEVE.

En momento de escribir este trabajo, los principales miembros de la fundación ONF, así como los principales operadores de servicios ya cuentan o están migrando sus centros de procesamiento de datos a soluciones SDN.

En definitiva, se puede afirmar que cada vez resulta más visible el uso y la utilización del software como medio para la gestión y el despliegue de las redes. La simbiosis existente entre SDN y NFV se propone como una potente alternativa a las técnicas de encaminamiento tradicionales, posibilitando introducir servicios de red personalizados de manera dinámica y eficiente para empezar a ganar mercado como tecnologías base para el estudio de las redes de un futuro próximo.

9.1 Lecciones aprendidas

A nivel personal, este trabajo ha supuesto el aprendizaje de múltiples herramientas y tecnologías que como profesional de las telecomunicaciones debería conocer. Ha completado mi curiosidad personal sobre las redes definidas por software y me ha aportado una visión más amplia sobre las posibilidades de negocio.

De entre las diferentes lecciones aprendidas y competencias adquiridas durante el desarrollo de este proyecto, cabe destacar el aprendizaje acerca de ONOS y OpenVSwitch.

En cuanto a las tecnologías analizadas, los conocimientos adquiridos son interesantes para cualquier rol a desempeñar dentro de un centro de procesamiento de datos a nivel de diseño o ejecución y sirven como base para alcanzar nuevos estudios o metas profesionales.

10. Valoración Productiva

Para realizar el cálculo de los costes del proyecto se han tenido en cuenta tanto los gastos personales como los materiales.

Con relación a los gastos personales, se parte de una jornada de 4h/día los 7 días de la semana durante los 3 meses y 16 días de realización del proyecto. Y con base al salario bruto mensual de un Técnico de Telecomunicaciones 1396 €/mes. [83]

Rol	Horas	Precio Hora	Coste diario
Técnico Telecomunicaciones	**4h/día**	**30 €**	**120 €**

En cuanto a los gastos materiales, se pueden dividir en gastos de Software y Hardware.

- **Software:** Este proyecto se ha realizado en base al software libre, por lo que no existen gastos por pago de licencias.

- **Hardware:** Se ha utilizado únicamente un portátil con conexión a internet. El despliegue de máquinas o laboratorios virtuales no suponen gastos de Hardware.

Hardware	Precio	Coste Hardware
Portátil MSI Bravo 17	**1.100 €**	**1.100 €**

El coste final del proyecto incluyendo los gatos personales y materiales es de:

Rol	Duración del proyecto	Coste Total
Técnico Telecomunicaciones	**109 días**	**14.180 €**

11. Glosario

API	Application Programming Interface
BW	Bandwith
CAPEX	Capital Expenditure
CDN	Content Delivery Network
COTS	Commercial Off-The-Shelf
CPD	Centro de Procesamiento de Datos
CSP	Communication Service Provider
DCN	Data Control Network
DPDK	Data Plane Development Kit
ECMP	Equal Cost Multi-Path
ENIAC	Electronic Numerical Integrator And Computer
ETSI	European Telecommunications Standards Institute
GGNAT	Carrier Grade NAT
IEEE	Institute of Electrical and Electronics Engineers
LAG	Link Aggregation Group
LISP	Cisco Locator ID Separation Protocol
MANO	Management and orchestration
MC-LAG	Multi-Chassis Link Aggregation Group
MEF	Metro Ethernet Forum
MIT	Massachusetts Institute of Technology
MSDC	Massively Scalable Data Center Network
NFV	Network functions virtualization
NFVO	Network functions virtualization Orchestrator
NIST	The National Institute of Standards and Technology
NVGRE	Generic Routing Encapsulation
OCP	Open Compute Project
ODL	Open DayLight
ONAP	Open Network Automation Platform
ONF	Open Network Fundation
ONOS	Open Network Operating System
OPEX	Operational expenditures
OTV	Overlay transport virtualization
PBB	Provider Backbone Bridge
POD	Point Of Delivery
RB	Router Bridge
SBC	Session Border Controller
SDN	Software-defined networking
SONA	Simplified Overlay Network Architecture
SPB	Shortest Path Bridging
SPBM	Shortest Path Bridging Multipath
SPBV	Shortest Path Bridging VID
STP	Spanning Tree Protocol
STT	Stateless Transport Tunneling
TCO	Total Cost Of Ownership

TOR	Top of Rack
TRADIC	TRAnsistor DIgital Computer
TRILL	Transparent Interconnection of Lots of Links
VM	Virtual Machine
VMM	Virtual Machine Monitor
VNFM	Virtualized Infrastructure Manager
VXLAN	Virtual Extensible LAN

12. Bibliografía

Lista numerada de las referencias bibliográficas utilizadas dentro de la memoria.

[1] (2022,29 septiembre). Statia.com. Recuperado de: https://es.statista.com/estadisticas/635666/prevision-trafico-mundial-en-internet-por-red-de-entrega-de-contenidos/

[2] David Reinse, (2018, noviembre). The Digitization of the World. Recuperado de: https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf

[3] (2022,29 septiembre). Metro Ethernet Forum. Recuperado de: https://www.mef.net/service-standards/overlay-services/sase/ y https://www.mef.net/service-standards/overlay-services/sd-wan/

[4] (2022,29 septiembre). IEEE.com. Recuperado de https://sdn.ieee.org/
[5] (2022,29 septiembre). Open Network Fundation. Recuperado de https://opennetworking.org/
[6] (2022,29 septiembre). PNETLAB.com Recuperado de https://www.pnetlab.com/pages/main

[7] D. Patterson, "50 Years of computer architecture: From the mainframe CPU to the domain-specific tpu and the open RISC-V instruction set," 2018 IEEE International Solid – State Circuits Conference – (ISSCC), 2018, pp. 27-31, doi: 10.1109/ISSCC.2018.8310168.

[8] J. W. Cortada, "The ENIAC's influence on business computing, 1940s-1950s," in IEEE Annals of the History of Computing, vol. 28, no. 2, pp. 26-28, April-June 2006, doi: 10.1109/MAHC.2006.38.

[10] L. C. Brown, "Flyable TRADIC: the first airborne transistorized digital computer," in IEEE Annals of the History of Computing, vol. 21, no. 4, pp. 55-61, oct.-Dec. 1999, doi: 10.1109/85.801533.

[11]J. Fong and C. Pottle, "Parallel Processing of Power System Analysis Problems Via Simple Parallel Microcomputer Structures," in IEEE Transactions on Power Apparatus and Systems, vol. PAS-97, no. 5, pp. 1834-1841, Sept. 1978, doi: 10.1109/TPAS.1978.354677.

[12] S. Medhat, "Informática cliente/servidor: un motor para el cambio y el crecimiento", Seminario internacional sobre informática cliente/servidor. Key Note Addresses, 1995, pp. 1/1-120 vol.2, doi: 10.1049/ic:19951146.

[13] Peleg, David (2000), Distributed Computing: A Locality-Sensitive Approach, SIAM, ISBN 978-0-89871-464-7

[14] M. Obali y AE Topcu, "Comparación de cluster, Grid y Cloud Computing usando tres enfoques diferentes", 23. ° Conferencia de aplicaciones de comunicaciones y procesamiento de señales (SIU) de 2015, 2015, xten. 192-195, doi: 10.1109/SIU.2015.7130446.

[15] S. K. Siu and J. Lopopolo, "Compatibility, Sizing, and Design Considerations for Generators and UPSs in Tiers I, II, III, and IV Topologies," in IEEE Transactions on Industry Applications, vol. 47, no. 6, pp. 2324-2329, nov.-Dec. 2011, doi: 10.1109/TIA.2011.2168594.

[16] (2022,2 octubre) IEEE.org. Recuperado de: https://www.ieee802.org/1/files/public/docs2009/aq-seaman-merged-spanning-tree-protocols-0509.pdf

[17] "IEEE Standard for Local and metropolitan xte networks–Link Aggregation," in IEEE Std 802.1AX-2008, vol., no., pp.1-163, 3 nov. 2008, doi: 10.1109/IEEESTD.2008.4668665.

[18] Juniper Networks. (2014). Recuperado de:

https://www.juniper.net/documentation/en_US/junose15.1/topics/concept/802.3ad-link-aggregation-understanding.html

[19] C. Clos, "A study of non-blocking switching networks," in The Bell System Technical Journal, vol. 32, no. 2, pp. 406-424, March 1953, doi: 10.1002/j.1538-7305. 1953.tb01433. x.
W. Sehery and C. Clancy, "Flow Optimization in Data Centers With Clos Networks in Support of Cloud Applications," in IEEE Transactions on Network and Service Management, vol. 14, no. 4, pp. 847-859, Dec. 2017, doi: 10.1109/TNSM.2017.2761321.

[20] M. Sultan, D. Imbuido, K. Patel, J. MacDonald and K. Ratnam, "Designing knowledge plane to optimize leaf and spine data center," 2020 IEEE 13th International Conference on Cloud Computing (CLOUD), 2020, pp. 13-15, doi: 10.1109/CLOUD49709.2020.00011.

[21] RFC 2992. (November,2000) Analysis of an Equal-Cost Multi-Path Algorithm. Recuperado de https://datatracker.ietf.org/doc/html/rfc2992

[22] C. E. Leiserson, "Fat-trees: Universal networks for hardware-efficient supercomputing," in IEEE Transactions on Computers, vol. C-34, no. 10, pp. 892-901, oct. 1985, doi: 10.1109/TC.1985.6312192.

[23-24] J. Alqahtani and B. Hamdaoui, "Rethinking Fat-Tree Topology Design for Cloud Data Centers," 2018 IEEE Global Communications Conference (GLOBECOM), 2018, pp. 1-6, doi: 10.1109/GLOCOM.2018.8647774.

[25] (2022,4 octubre). Amazon.com. Recuperado de:
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html

G. Minutoli, M. Fazio, M. Paone and A. Puliafito, "Virtual business networks with Cloud Computing and virtual machines," 2009 International Conference on Ultra Modern Telecommunications & Workshops, 2009, pp. 1-6, doi: 10.1109/ICUMT.2009.5345440.

[26] (2022,4 octubre). NIST. Recuperado de:
https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

[27] POPEK, Gerald J.; GOLDBERG, Robert P. Formal requirements for virtualizable third generation architectures. Communications of the ACM, 1974, vol. 17, no 7, p. 412-421.

[28] S. Kolhe and S. Dhage, "Comparative study on Virtual Machine Monitors for cloud," 2012 World Congress on Information and Communication Technologies, 2012, pp. 425-430, doi: 10.1109/WICT.2012.6409115

[29] G. Kaur and V. Grover, "Detection and Prevention of Hypervisor and VM Attacks," 2021 IEEE International Conference on Nanoelectronics, Nanophotonics, Nanomaterials, Nanobioscience & Nanotechnology (5NANO), 2021, pp. 1-5, doi: 10.1109/5NANO51638.2021.9491137.

[30] (2022,5 octubre). Open Compute Project. Recuperado de https://www.opencompute.org/

[31] J. Kon, N. Mizusawa, A. Umezawa, S. Yamaguchi and J. Tao, "Highly consolidated servers with container-based virtualization," 2017 IEEE International Conference on Big Data (Big Data), 2017, pp. 2472-2479, doi: 10.1109/BigData.2017.8258205.

[32] Libro: Network Function Virtualization: Concepts and Applicability in 5G Networks. Autor: Ying Zhang. Año:2018

[33]S. Kolhe and S. Dhage, "Comparative study on Virtual Machine Monitors for cloud," 2012 World Congress on Information and Communication Technologies, 2012, pp. 425-430, doi: 10.1109/WICT.2012.6409115.

[34] (2022,10 octubre) ETSI.ORG. Recuperado de https://www.etsi.org/technologies/nfv

[35] (2022,10 octubre) LINUX FUNDATION.Recuperado de https://www.dpdk.org/

[36] R. R. Reyes, S. Sultana, V. V. Pai and T. Bauschert, "Analysis and Evaluation of CAPEX and OPEX in Intra-Data Centre Network Architectures," 2019 IEEE Latin-American Conference on Communications (LATINCOM), 2019, pp. 1-6, doi: 10.1109/LATINCOM48065.2019.8937881.

[37] ISGN12: ISG NFV. Network Functions Virtualization: An Introduction, Benefits, Enablers, Challenges & Call for Action. ISG NFV White Paper, October 2012.

[38] (2022,11 octubre) WIKIPEDIA.Recuperado de https://en.wikipedia.org/wiki/FCAPS
[39] (2022,11 octubre) OPEN MANO Recuperado de https://osm.etsi.org/

[40] N. –F. Huang, C. –H. Li, C. –C. Chen, I. –H. Hsu, C. –C. Li and C. –H. Chen, "A novel vCPE framework for enabling virtual network functions with multiple flow tables architecture in SDN switches," 2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS), 2017, pp. 64-69, doi: 10.1109/APNOMS.2017.8094180.

[41-42-43] Guy Pujolle, "Fabric, SD-WAN, vCPE, vRAN, vEPC," in Software Networks: Virtualization, SDN, 5G, and Security, Wiley, 2020, pp.33-50, doi: 10.1002/9781119694748.ch3.

[44] (2022,15 octubre) OPNFV Recuperado de https://www.opnfv.org/
[45] (2022,15 octubre) ONAP Recuperado de https://www.onap.org
F. Slim, F. Guillemin, A. Gravey and Y. Hadjadj-Aoul, "Towards a dynamic adaptive placement of virtual network functions under ONAP," 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2017, pp. 210-215, doi: 10.1109/NFV-SDN.2017.8169880.

[46] (2017) ECOMP (Enhanced Control, Orchestration, Management & Policy) Architecture White Paper. Recuperado de https://about.att.com/content/dam/snrdocs/ecomp.pdf

[47] Universidad de Princeton y Carnegie Mellon. Design and Implementation of a Routing Control Platform.Recuperado de https://www.cs.princeton.edu/~jrex/papers/rcp-nsdi.pdf

[48] Universidad de Stanford y Berkeley.SANE: A Protection Architecture for Enterprise Networks.Recuperado de http://yuba.stanford.edu/~casado/sane/index.html

[49] (2012) OPEN SUMMIT SDN. Recuperado de:
https://www.networkworld.com/article/2222187/open-networking-summit-2012-day-1-recap--part-1-.html

[50] (2014) ONF Recuperado de:
https://www.ramonmillan.com/documentos/bibliografia/OpenFlowEnabledSDN&NFV_ONF.pdf

[51] Natarajan, Sriram; et al. (2013). "A Software defined Cloud-Gateway automation system using OpenFlow". 2013 IEEE 2nd International Conference on Cloud Networking (Cloud Net). IEEE Xplore. Pp. 219–226. Doi:10.1109/CloudNet.2013.6710582. ISBN 978-1-4799-0568-3.

[52] A Study on the Dependability of Software Defined Networks – Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/ndicates-the-architecture-of-a-SDN-which-includes-three-layers-and-two-APIs-The-three_fig2_301431575 [accessed 10 Nov, 2022]

[53] (2022,20 octubre) REST APIs: How They Work and What You Need to Know Recuperado de https://blog.hubspot.com/website/what-is-rest-api

[54] (2022,20 octubre) ODL Recuperado de https://www.opendaylight.org/

[55] (2022,20 octubre) ONOS Recuperado de:
https://wiki.onosproject.org/display/ONOS/Guides

[56] (2022,20 octubre) TUGNSTEN FABRIC Recuperado de:
https://tungstenfabric.github.io/website/Tungsten-Fabric-Architecture.html

[57] (2022,20 octubre) HP VAN Recuperado de: https://techhub.hpe.com/eginfolib/networking/docs/sdn/sdnc2_6/5998-8473install/content/c_overview.html

[58] (2022,20 octubre) NEC PF6800 Recuperado de: https://www.necam.com/doclibrary/NEC_ProgrammableFlow_SDN_OpenFlow-based_%20PF6800_PFTAP_User_Guide.pdf

[59] (2022,20 octubre) VMWARE NSX Recuperado de: https://docs.vmware.com/en/VMware-NSX-Data-Center-for-vSphere/6.4/com.vmware.nsx.admin.doc/GUID-B5C70003-8194-4EC3-AB36-54C848508818.html

[60] (2022,20 octubre) NOKIA NUAGE Recuperado de https://documentation.nokia.com/cgi-bin/dbaccessfilename.cgi/3HE18121AAAATQZZA_V1_NSP%2022.3%20User%20Guide.pdf

[61] (2022,20 octubre) CISCO VIPTELA Recuperado de https://sdwan-docs.cisco.com/Product_Documentation/Getting_Started

[62] (2022,21 octubre) MSDC WHITE PAPER Recuperado de https://www.cisco.com/c/en/us/products/collateral/switches/nexus-9000-series-switches/white-paper-c11-743245.html

[63] Libro: Using TRILL, FabricPath, and VXLAN.Autor: Sanjay K. Hooda, Shyam Kapadia, Padmanabhan Krishnan.Año:2014
Libro: SDN: Software Defined Networks.Autor: Thomas D. Nadeau and Ken Gray. Año:2013

[64-65] Libro: Software Networks.Autor: Guy Pujolle Año:2015

[66] “ISO/IEC/IEEE International Standard – Telecommunications and xtensió between information technology systems — Requirements for local and metropolitan xte networks —Part 1Q: Bridges and bridged networks AMENDMENT 3: Virtual station interface (VSI) xtensión and configuration protocol (VDP) xtensión to support network virtualization overlays over layer 3 (NVO3),” in ISO/IEC/IEEE 8802-1Q:2020/Amd.3:2021€, vol., no., pp.1-42, 22 Sept. 2021, doi: 10.1109/IEEESTD.2021.9546720.
U. Amirsaidov, N. Usmanova and B. Samandarov, “Overlay Networking Issues: Implementation The Functional Components,” 2020 International Conference on Information Science and Communications Technologies (ICISCT), 2020, pp. 1-4, doi: 10.1109/ICISCT50599.2020.9351404.
Harnessing Complex Structures and Collective Dynamics in Large Networked Computing Systems – Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Example-Overlay-Network-built-on-top-of-an-Internet-style-Underlay_fig4_230774628 [accessed 11 Nov, 2022]

[67] (2022,25 octubre) NVE Recuperado de https://www.ietf.org/proceedings/83/slides/slides-83-nvo3-4.pdf

[68] (2022,29 junio) FS Community –Switches ToR y ToRs populares en arquitecturas de centros de datos Recuperado de https://community.fs.com/es/blog/popular-tor-and-tor-switch-in-data-center-architectures.html

[69] (2022,26 octubre) HUAWEI SUPPORT Hybrid Overlay Recuperado de: https://support.huawei.com/enterprise/en/doc/EDOC1100072313/1b4b1d32/hybrid-overlay

[70] (2022,26 octubre) 802.1ah MAC-IN-MAC Recuperado de: https://documentation.nokia.com/html/0_add-h-f/93-0076-10-01/7750_SR_OS_Services_Guide/services_PBB.html

[71] (2022,26 octubre) CISCO FORUM Recueprado de: https://learningnetwork.cisco.com/s/question/0D53i00000Ksr2nCAB/fabric-technologies-and-overlays

[72] Perlman, R., Eastlake 3rd, D., Dutt, D., Gai, S., and A. Ghanwani, “Routing Bridges (Rbridges): Base Protocol Specification”, RFC 6325, DOI 10.17487/RFC6325, July 2011, <https://www.rfc-editor.org/info/rfc6325>.

TRILL Technology White Paper Recuperado de https://www.h3c.com/en/Products_Technology/Operating_System/ComwareV7/Data_Center/White_Papers/201808/1102744_294549_0.htm

[73] IS-IS Extensions Supporting IEEE 802.1aq Shortest Path Bridging Recuperado de https://datatracker.ietf.org/doc/rfc6329/

(2022,27 octubre) Recuperado de: https://archive.nanog.org/meetings/nanog50/presentations/Monday/NANOG50.Talk63.NANOG50_TRILL-SPB-Debate-Roisman.pdf

[74] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <https://www.rfc-editor.org/info/rfc7348>.
Libro: Building Data Centers with VXLAN BGP EVPN: A Cisco NX-OS Perspective.Autor: Lukas Krattiger, Shyam Kapadia, David Jansen.Año:2017

[75] Sajassi, A., Ed., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February 2015, <https://www.rfc-editor.org/info/rfc7432>.
Libro: EVPN in the Data Center.Autor: Dinesh G. Dutt. Año:2018

[76] Garg, P., Ed., and Y. Wang, Ed., "NVGRE: Network Virtualization Using Generic Routing Encapsulation", RFC 7637, DOI 10.17487/RFC7637, September 2015, <https://www.rfc-editor.org/info/rfc7637>.
[77] (2022,28 Octubre) MICROSOFT LEARNING Recuperado de https://learn.microsoft.com/en-us/windows-server/networking/sdn/technologies/hyper-v-network-virtualization/hyperv-network-virtualization-technical-details-windows-server

[78] October 30, 2016 Vmware, B. Davie, Ed. Inc.A Stateless Transport Tunneling Protocol for Network Virtualization (STT) draft-davie-stt-08 Network Working Group IETF

[79] Gross, J., Ed., Ganga, I., Ed., and T. Sridhar, Ed., "Geneve: Generic Network Virtualization Encapsulation", RFC 8926, DOI 10.17487/RFC8926, November 2020, <https://www.rfc-editor.org/info/rfc8926>.

[80] (2022,28 octubre) Recuperado de: https://www.cisco.com/c/dam/en_us/solutions/industries/docs/gov/otv.pdf
August 27, 2013 Cisco Systems Overlay Transport Virtualization draft-hasmit-otv-04 Recuperado de https://datatracker.ietf.org/doc/html/draft-hasmit-otv-04

[81] Meyer, D., Ed., Zhang, L., Ed., and K. Fall, Ed., "Report from the IAB Workshop on Routing and Addressing", RFC 4984, DOI 10.17487/RFC4984, September 2007, <https://www.rfc-editor.org/info/rfc4984>.

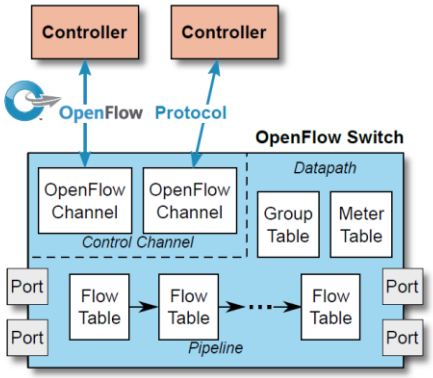
[82] Libro: LISP Network, The: Evolution to the Next-Generation of Data Networks (Networking Technology) Autor: Farinacci, Dino; Moreno, Victor.Año:2017

[83] (2022,12 de Diciembre) Recuperado de: https://es.indeed.com/career/t%C3%A9cnico-en-telecomunicaciones/salaries

12. Anexos

12.1 Mensajes openflow:

OpenFlow cuenta con una gran cantidad de mensajes que le permiten transmitir acciones e informacion desde el controlador hacia los dispositivos.



Se clasifican en:

Mensajes del controlador al conmutador: Se generan en el controlador, y el switch necesariamente responde al mensaje. En esta categoría se encuentran los siguientes mensajes:

Features: Se envía cuando el controlador requiere obtener las características del switch, que le responde. Se usa normalmente en el establecimiento de una conexión OpenFlow.

Configuration: Mensajes de consulta de los parámetros de configuración del switch.

Modify-State: Se envía desde el controlador para la gestión de estados en el switch, como por ejemplo añadir o quitar flujos o cambiar el estado de un determinado puerto.

Read-State: El controlador lo usa para adquirir las características del switch.

Packet-Out: El controlador los usa para enviar paquetes por un determinado puerto del switch y para redireccionar paquetes Packet-In. Encapsula el paquete a ser redireccionado y las acciones que se aplicarán al mismo.

Barriel: El controlador lo usa para asegurarse que las dependencias de un mensaje se han cumplido o para recibir notificaciones por operaciones finalizadas.

Mensajes asíncronos: Estos mensajes se envían desde los switches al controlador, cuando un paquete llega, cambio de estado tras un error. En este tipo de paquetes están:

Packet In: Paquete enviado desde el switch hacia el controlador cuando no tiene una entrada en su tabla de flujos que concuerde con el paquete entrante. El controlador procesa el paquete y responde con un mensaje ***Packet-Out***.

Flow-Removed: Se envía cuando el tiempo de inactividad de un flujo suspira. Puede ser enviado tanto por el controlador como por el switch.

Port-Status: Se envía del switch al controlador cuando la configuración de un puerto cambia.

Error: El switch notifica al controlador si existen problemas con estos mensajes.

Mensajes síncronos: Estos mensajes se envían desde cualquier lugar, controlador o dispositivo sin solicitud previa. En esta clasificación se encuentra los siguientes mensajes:
Hello: Estos mensajes se intercambian al momento del establecimiento de la conexión entre los switches y el controlador.

Echo: Se usan para medir la latencia o el ancho de banda o para verificar que un dispositivo esté activo. Por cada petición que llegue al destino, se creará una respuesta que será enviada al origen. Puede ser enviado por el switch o el controlador.

Experimenter: Permite dar funcionalidades adicionales a un switch OpenFlow, habiéndose planeado para futuras versiones del protocolo.

Tabla de Flujo:

Las entradas en una tabla de flujo son combinaciones flexibles de ciertas palabras clave y acciones. Cada entrada de flujo en una tabla consta de campos de coincidencia y un conjunto de instrucciones que se aplican a los paquetes coincidentes. Al recibir un paquete de datos, un switch OpenFlow analiza y compara el encabezado del paquete con los campos de coincidencia en las entradas de flujo y ejecuta la instrucción correspondiente si se encuentra una coincidencia. Su estructura de entrada de flujo varía según las versiones de OpenFlow,

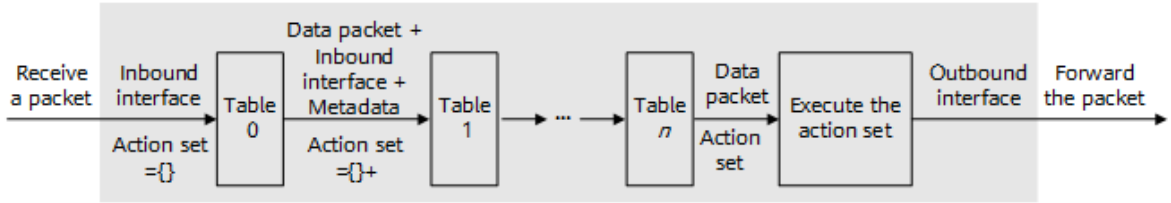


Ilustración 82- Procesamiento Tabla de Flujo

Las tablas de flujo de OpenFlow se pueden entregar en modo proactivo o reactivo.

- ***Modo proactivo:*** El controlador entrega la información de la tabla de flujo a los switches OpenFlow.

- ***Modo reactivo:*** Cuando un switch OpenFlow no encuentra ninguna entrada de flujo que coincida con un paquete recibido, envía un mensaje al controlador. Luego, el controlador determina cómo reenviar el paquete y calcula y entrega la entrada de flujo correspondiente al Switch. De este modo, el switch no necesita mantener todas las entradas de flujo. En su lugar, obtiene entradas de flujo del controlador y las

guarda solo para el tráfico realmente generado, y elimina las entradas de flujo cuando caducan

12.2 Configuraciones Previas Despliegue LAB1

12.2.1 Instalación Mininet:

Para confirmar qué versión del sistema operativo está ejecutando, ejecute el comando

```
lsb_release -a

sudo apt-get install mininet
```

Test Mininet:

```
mn --version
sudo mn --switch ovsbr --test pingall
```

Si Mininet se queja de que Open vSwitch no funciona

```
sudo apt-get install openvswitch-switch
sudo service openvswitch-switch start
```

Instalará el switch de referencia OpenFlow, el controlador de referencia y el analizador Wireshark.

```
mininet/util/install.sh -fw
```

Las imágenes oficiales de Mininet VM incluyen Python 2 y Python 3 Mininet.Cuando instala desde la fuente, puede elegir qué versión instalar. Por ejemplo:

```
sudo PYTHON=python2 mininet/util/install.sh -n   # install Python 2 Mininet
sudo PYTHON=python3 mininet/util/install.sh -n   # install Python 3 Mininet
```

12.2.2 Instalación Java 11:

```
sudo apt install openjdk-11-jdk
```

```
sudo su
cat >> /etc/environment <<EOL
JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
JRE_HOME=/usr/lib/jvm/java-11-openjdk-amd64/jre
EOL
```

12.2.3 Instalación Maven:

```
sudo apt update
sudo apt-get install maven
mvn --version
```

12.2.4 Instalación Karaf:

```
wget https://archive.apache.org/dist/karaf/3.0.5/apache-karaf-3.0.5-src.tar.gz

tar -zxvf apache-karaf-3.0.5.tar.gz -C

wget http://apt.puppetlabs.com/puppetlabs-release-precise.deb

sudo dpkg -i puppetlabs-release-precise.deb

sudo apt-get install gcc g++ maje libpcre3-dev libssl-dev tmux emacs libncurses5-dev
libreadline-dev libffi-dev libyaml-dev valgrind git-core libxml2-dev libxslt-dev ntp
sqlite3 libsqgit clo lite3-dev
```

12.2.5 Instalación Git Core:

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

12.2.6 Instalación Curl:

```
sudo apt-get install curl
```

12.2.7 Instalación Docker y Atomix:

```
sudo apt install docker.io
```

1. Descargue la imagen de la ventana acoplable Atomix:

```
$ docker pull atomix/atomix:3.1.5
```

2. Ejecute tres instancias de Atomix:

```
$ docker run -t -d --name atomix-1 atomix/atomix:3.1.5

$ docker run -t -d --name atomix-2 atomix/atomix:3.1.5

$ docker run -t -d --name atomix-3 atomix/atomix:3.1.5
```

3. Verifique la IP de la ventana acoplable de las instancias de Atomix;

```
$ docker inspect atomix-1 | grep -i ipaddress

$ docker inspect atomix-2 | grep -i ipaddress

$ docker inspect atomix-3 | grep -i ipaddress
```

4. Consulte el código fuente de ONOS:

```
$ git clone https://gerrit.onosproject.org/onos
```

5. Configure las variables de entorno relevantes para la IP de la ventana acoplable de las instancias de Atomix obtenidas anteriormente:

```
$ export OC1=172.17.0.5

$ export OC2=172.17.0.6

$ export OC3=172.17.0.7
```

6. Genere archivos de configuración de Atomix:

```
cd /path/to/your/onos/source/root

./tools/test/bin/atomix-gen-config  172.17.0.5  ~/atomix-1.conf  172.17.0.5  172.17.0.6
172.17.0.7

./tools/test/bin/atomix-gen-config  172.17.0.6  ~/atomix-2.conf  172.17.0.5  172.17.0.6
172.17.0.7

./tools/test/bin/atomix-gen-config  172.17.0.7  ~/atomix-3.conf  172.17.0.5  172.17.0.6
172.17.0.7
```

7. Copie la configuración de Atomix en las instancias de Docker:

```
$ docker cp ~/atomix-1.conf atomix-1:/opt/atomix/conf/atomix.conf

$ docker cp ~/atomix-2.conf atomix-2:/opt/atomix/conf/atomix.conf

$ docker cp ~/atomix-3.conf atomix-3:/opt/atomix/conf/atomix.conf
```

8. Reinicie las instancias de docker de Atomix para que la configuración surta efecto:

```
$ docker restart atomix-1

$ docker restart atomix-2

$ docker restart atomix-3
```

9. Descargue la imagen de la ventana acoplable ONOS:

```
$ docker pull onosproject/onos:1.15.0
```

10. Ejecute tres instancias de docker de ONOS:

```
$ docker run -t -d --name onos1 onosproject/onos:1.15.0

$ docker run -t -d --name onos2 onosproject/onos:1.15.0

$ docker run -t -d --name onos3 onosproject/onos:1.15.0
```

11. Verifique la IP de la ventana acoplable de las instancias de ONOS;

```
$ docker inspect onos1 | grep -i ipaddress

$ docker inspect onos2 | grep -i ipaddress

$ docker inspect onos3 | grep -i ipaddress
```

12. Genere los archivos de configuración del clúster de ONOS utilizando la IP de la ventana acoplable obtenida anteriormente:

```
cd /path/to/your/onos/source/root

./tools/test/bin/onos-gen-config 172.17.0.5 ~/cluster-1.json -n 172.17.0.5 172.17.0.6
172.17.0.7

./tools/test/bin/onos-gen-config 172.17.0.6 ~/cluster-2.json -n 172.17.0.5 172.17.0.6
172.17.0.7

./tools/test/bin/onos-gen-config 172.17.0.7 ~/cluster-3.json -n 172.17.0.5 172.17.0.6
172.17.0.7
```

13. Cree el directorio de configuración para las instancias de la ventana acoplable ONOS:

```
$ docker exec onos1 mkdir /root/onos/config

$ docker exec onos2 mkdir /root/onos/config

$ docker exec onos3 mkdir /root/onos/config
```

14. Copie la configuración del clúster de ONOS en las instancias de Docker:

```
$ docker cp ~/cluster-1. json onos1:/root/onos/config/cluster.json

$ docker cp ~/cluster-2.json onos2:/root/onos/config/cluster.json

$ docker cp ~/cluster-3.json onos3:/root/onos/config/cluster.json
```

15. Reinicie las instancias de la ventana acoplable ONOS para que la configuración surta efecto:

```
$ docker restart onos1

$ docker restart onos2

$ docker restart onos3
```

12.2.8 Configuración Atomix y Cluster Onos:

- ***Atomix-1***

```
cluster {
  cluster-id: onos
  node.id: atomix-1
  node.address: "172.17.0.2:5679"
  discovery {
    type: bootstrap
    nodes.1 {
      id: atomix-1
      address: "172.17.0.2:5679"
    }
    nodes.2 {
      id: atomix-2
      address: "172.17.0.3:5679"
    }
    nodes.3 {
      id: atomix-3
      address: "172.17.0.4:5679"
    }
  }
}

management-group {
  type: raft
  partitions: 1
  storage.level: disk
  members: [atomix-1, atomix-2, atomix-3]
}

partition-groups.raft {
  type: raft
  partitions: 7
  storage.level: disk
```

```
  members: [atomix-1, atomix-2, atomix-3]
}
```

- ***Atomix-2***

```
cluster {
  cluster-id: onos
  node.id: atomix-2
  node.address: "172.17.0.3:5679"
  discovery {
    type: bootstrap
    nodes.1 {
      id: atomix-1
      address: "172.17.0.2:5679"
    }
    nodes.2 {
      id: atomix-2
      address: "172.17.0.3:5679"
    }
    nodes.3 {
      id: atomix-3
      address: "172.17.0.4:5679"
    }
  }
}

management-group {
  type: raft
  partitions: 1
  storage.level: disk
  members: [atomix-1, atomix-2, atomix-3]
}

partition-groups.raft {
  type: raft
  partitions: 7
  storage.level: disk
  members: [atomix-1, atomix-2, atomix-3]
}
```

- ***Atomix-3***

```
cluster {
  cluster-id: onos
  node.id: atomix-3
  node.address: "172.17.0.4:5679"
  discovery {
    type: bootstrap
    nodes.1 {
      id: atomix-1
      address: "172.17.0.2:5679"
    }
    nodes.2 {
      id: atomix-2
      address: "172.17.0.3:5679"
    }
    nodes.3 {
      id: atomix-3
      address: "172.17.0.4:5679"
    }
  }
```

```
}

management-group {
  type: raft
  partitions: 1
  storage.level: disk
  members: [atomix-1, atomix-2, atomix-3]
}

partition-groups.raft {
  type: raft
  partitions: 7
  storage.level: disk
  members: [atomix-1, atomix-2, atomix-3]
}
```

- ***Cluster-1.json***

```
{
  "node": {
    "ip": "172.17.0.5",
    "id": "172.17.0.5",
    "port": 9876
  },
  "storage": [
    {
      "ip": "172.17.0.2",
      "id": "atomix-1",
      "port": 5679
    },
    {
      "ip": "172.17.0.3",
      "id": "atomix-2",
      "port": 5679
    },
    {
      "ip": "172.17.0.4",
      "id": "atomix-3",
      "port": 5679
    }
  ],
  "name": "onos”
}
```

- ***Cluster-2.json***

```
{
  "node": {
    "ip": "172.17.0.6",
    "id": "172.17.0.6",
    "port": 9876
  },
  "storage": [
    {
      "ip": "172.17.0.2",
      "id": "atomix-1",
      "port": 5679
    },
    {
      "ip": "172.17.0.3",
      "id": "atomix-2",
      "port": 5679
    },
```

```
    {
      "ip": "172.17.0.4",
      "id": "atomix-3",
      "port": 5679
    }
  ],
  "name": "onos"
}
```

- ***Cluster-3.json***

```
{
  "node": {
    "ip": "172.17.0.7",
    "id": "172.17.0.7",
    "port": 9876
  },
  "storage": [
    {
      "ip": "172.17.0.2",
      "id": "atomix-1",
      "port": 5679
    },
    {
      "ip": "172.17.0.3",
      "id": "atomix-2",
      "port": 5679
    },
    {
      "ip": "172.17.0.4",
      "id": "atomix-3",
      "port": 5679
    }
  ],
  "name": "onos"
}
```

Las configuraciones se han realizado siguiendo las indicaciones del proyecto Onos de la página web: https://wiki.onosproject.org/display/ONOS/Forming+a+cluster y adaptado el código según la necesidad el proyecto.

12.2.9 Instalación Onos:

```
sudo mkdir /opt
cd /opt
```

```
sudo wget -c https://repo1.maven.org/maven2/org/onosproject/onos-
releases/1.15.0/onos-1.15.0.tar.gz

sudo tar xzf onos-1.15.0.tar.gz
sudo mv onos-1.15.0 onos
```

```
# COMPROBAR SERVICIO
```

```
sudo ./service-onos {start|stop|status}
```

```
# ACCESO WEB
```

```
http://172.17.0.X:8181/onos/ui/index.html
```

12.3 Configuraciones Despliegue LAB2

12.3.1 Instalación versión 6 Kernel:

```
$ wget -c https://kernel.ubuntu.com/~kernel-ppa/mainline/v6.0.1/amd64/linux-headers-6.0.1-060001_6.0.1-060001.202210120833_all.deb
$ wget -c https://kernel.ubuntu.com/~kernel-ppa/mainline/v6.0.1/amd64/linux-headers-6.0.1-060001-generic_6.0.1-060001.202210120833_amd64.deb
$ wget -c https://kernel.ubuntu.com/~kernel-ppa/mainline/v6.0.1/amd64/linux-image-unsigned-6.0.1-060001-generic_6.0.1-060001.202210120833_amd64.deb
$ wget -c https://kernel.ubuntu.com/~kernel-ppa/mainline/v6.0.1/amd64/linux-modules-6.0.1-060001-generic_6.0.1-060001.202210120833_amd64.deb
sudo dpkg -i linux-headers-6.0.1*.deb linux-modules-6.0.1*.deb linux-image-6.0.1*.deb
```

12.3.2 Instalación OpenvSwitch:

```
sudo apt-get install openvswitch-switch-dpdk
```

CONFIGURACION EN VM1:

```
sudo ip netns add vm1
sudo ip link add name veth1 type veth peer name sw1-p1
sudo ip link set dev veth1 netns vm1
sudo ip netns exec vm1 ifconfig veth1 10.10.0.1/24 up
sudo ovs-vsctl add-br sw1
sudo ovs-vsctl add-port sw1 sw1-p1
sudo ip link set sw1-p1 up
sudo ip link set sw1 up
```

#CREACCION DE TUNEL VXLAN

```
sudo ovs-vsctl add-port sw1 tun0 -- set interface tun0 type=vxlan options:remote_ip=192.168.70.13 options:key=uoc
```

#CREACCION DE TUNEL GENEVE

```
sudo ovs-vsctl add-port sw1 tun0 -- set interface tun0 type=geneve options:remote_ip=192.168.80.12 options:key=uoc
```

CONFIGURACION EN VM2:

```
sudo ip netns add vm2
sudo ip link add name veth1 type veth peer name sw2-p1
sudo ip link set dev veth1 netns vm2
sudo ip netns exec vm2 ifconfig veth1 10.10.0.2/24 up
sudo ovs-vsctl add-br sw2
sudo ovs-vsctl add-port sw2 sw2-p1
sudo ip link set sw2-p1 up
sudo ip link set sw2 up
```

#CREACCION DE TUNEL VXLAN

```
sudo ovs-vsctl add-port sw2 tun0 -- set interface tun0 type=vxlan options:remote_ip=192.168.70.11 options:key=uoc
```

#CREACCION DE TUNEL GENEVE

```
sudo ovs-vsctl add-port sw2 tun0 -- set interface tun0 type=geneve
options:remote_ip=192.168.80.11 options:key=uoc
```

 # TEST en VM1 y VM2

```
sudo ip netns exec vmX ping 10.10.0.X
```

```
sudo ovs-ofctl dump-flows swX
```

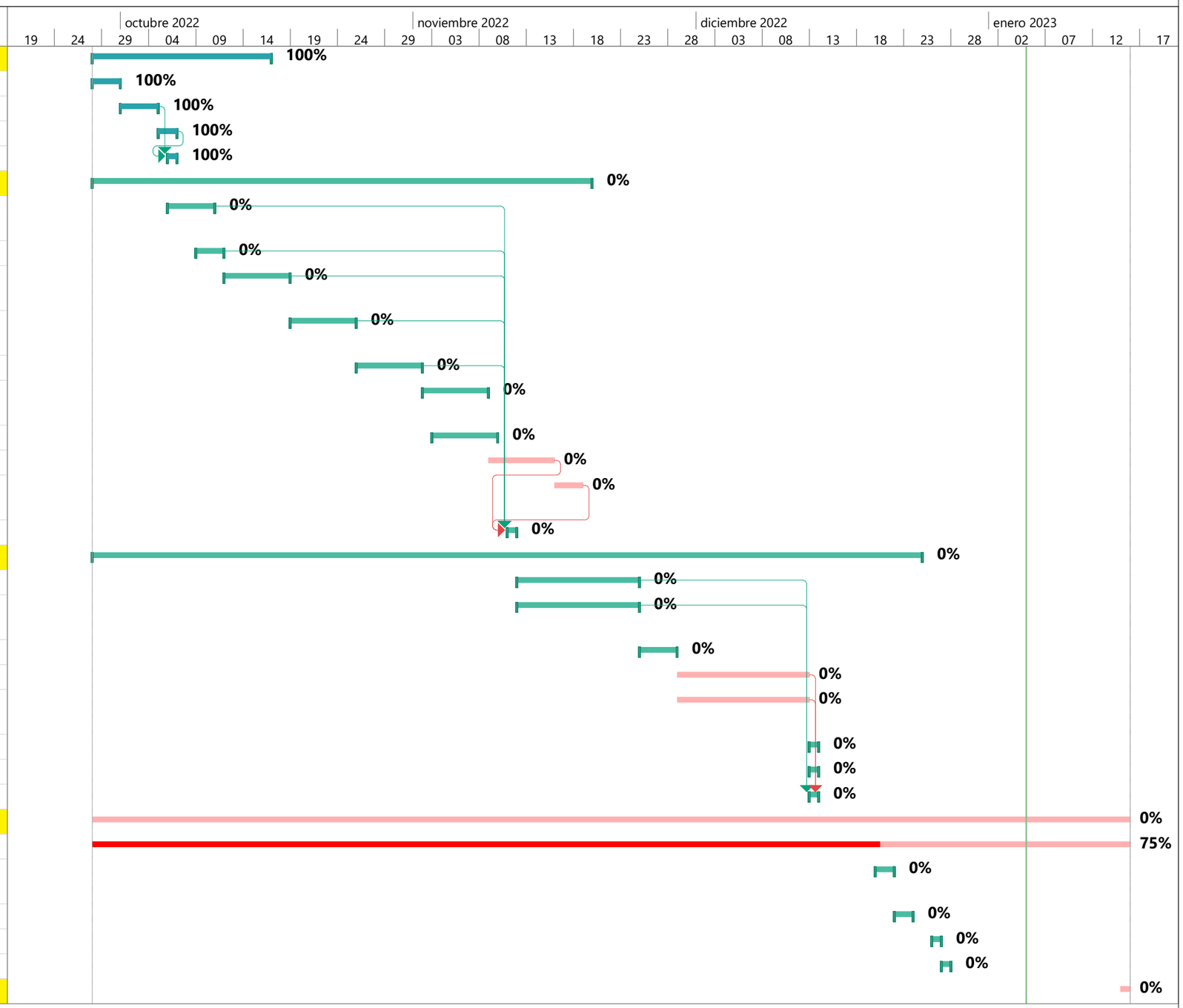
```
sudo ovs-vsctl show | swX
```

```
sudo ip netns exec vmX ifconfig
```

```
sudo tcpdump –nli swx-port1
```

Diagrama Gantt Inicial

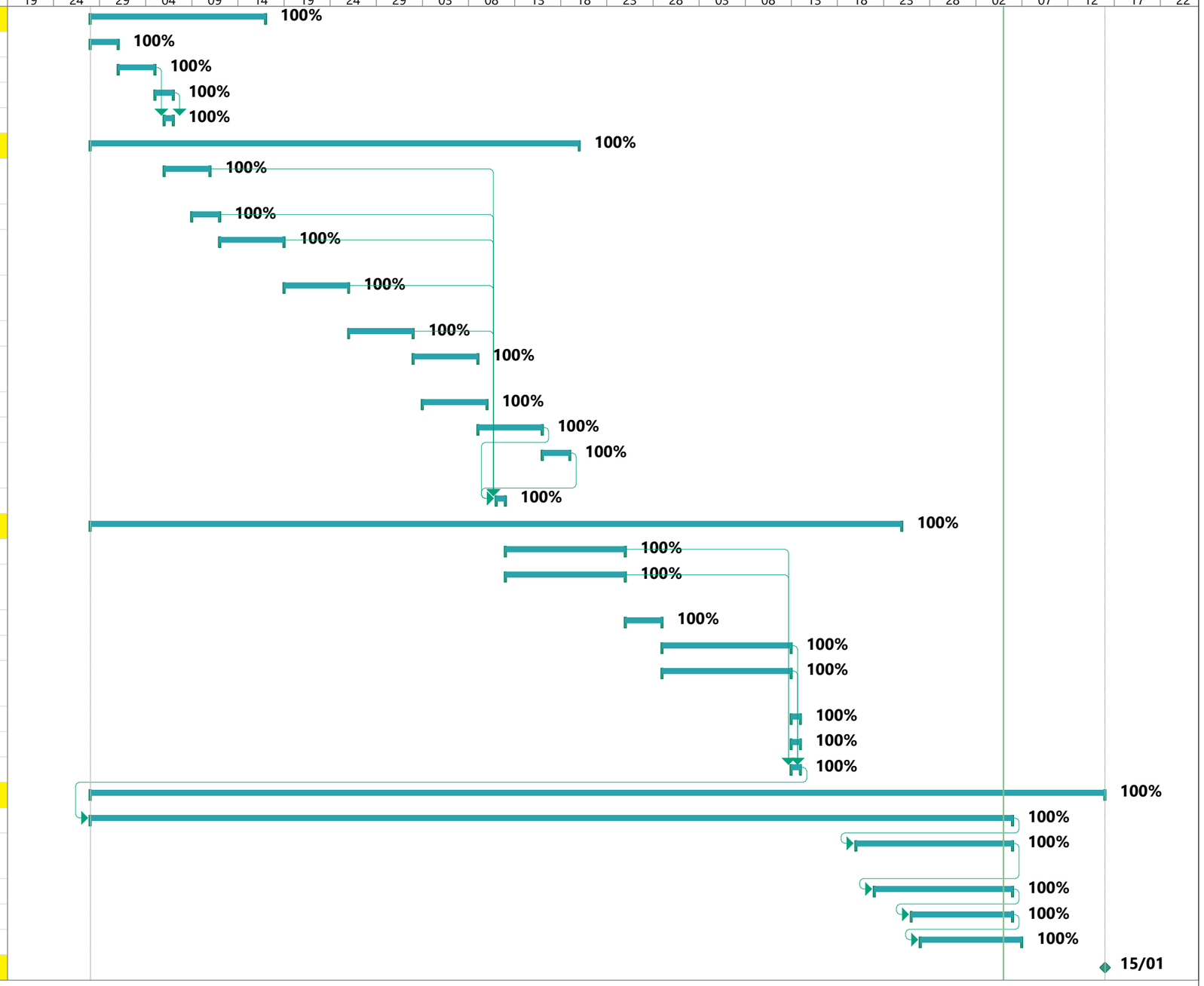
Id	Modo de tarea	Nombre de tarea	Duración	Comienzo
1		**PEC 1**	**14 días**	**mié 28/09/22**
2		Planteamiento	3 días	mié 28/09/22
3		Planificación	3 días	sáb 01/10/22
4		Cronograma	2 días	mié 05/10/22
5		Entrega PEC 1	1 día	jue 06/10/22
6		**PEC 2**	**39 días**	**mié 28/09/22**
7		Contexto y justificación del trabajo	3 días	jue 06/10/22
8		Objetivos y enfoque	3 días	dom 09/10/22
9		Documentación sobre arquitecturas de CPD	5 días	mié 12/10/22
10		Documentación sobre Cloud y virtualización	5 días	mié 19/10/22
11		Documentación sobre SDN	5 días	mié 26/10/22
12		Análisis de tecnologías y protocolos	5 días	mié 02/11/22
13		Análisis de controladores SDN	5 días	jue 03/11/22
14		Documentación sobre NFV	5 días	mié 09/11/22
15		Análisis sobre software a utilizar	3 días	mié 16/11/22
16		Entrega PEC 2	1 día	vie 11/11/22
17		**PEC 3**	**64 días**	**mié 28/09/22**
18		Creación de Laboratorio 1	10 días	sáb 12/11/22
19		Implementación de controlador SDN en ONOS	10 días	sáb 12/11/22
20		Resultados	2 días	vie 25/11/22
21		Creación de Laboratorio 2	10 días	mar 29/11/22
22		Implementación de tecnologías en OpenVswitch	10 días	mar 29/11/22
23		Resultados	1 día	mar 13/12/22
24		Valoración Productiva	1 día	mar 13/12/22
25		Entrega PEC 3	1 día	mar 13/12/22
26		**ENTREGA FINAL**	**79 días**	**mié 28/09/22**
27		Redacción de la memoria final	79 días	mié 28/09/22
28		Revisión u corrección de la memoria final	2 días	mar 20/12/22
29		Power Point	2 días	jue 22/12/22
30		Video Presentación	1 día	lun 26/12/22
31		Informe de autoevaluación	1 día	mar 27/12/22
32		**Final del TFG**	**1 día**	**dom 15/01/23**



Tareas críticas
División crítica
Progreso de tarea crítica
Tarea
División
Progreso de tarea
Tarea manual
solo el comienzo
solo fin
solo duración
Línea base
División de la línea base
Hito de línea base
Hito
Progreso del resumen
Resumen
Resumen manual
Resumen del proyecto
Tareas externas
Hito externo
Tarea inactiva
Hito inactivo
Resumen inactivo
Fecha límite

Diagrama Gantt Final

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo
1		**PEC 1**	**14 días**	**mié 28/09/22**
2		Planteamiento	3 días	mié 28/09/22
3		Planificación	3 días	sáb 01/10/22
4		Cronograma	2 días	mié 05/10/22
5		Entrega PEC 1	1 día	jue 06/10/22
6		**PEC 2**	**39 días**	**mié 28/09/22**
7		Contexto y justificación del trabajo	3 días	jue 06/10/22
8		Objetivos y enfoque	3 días	dom 09/10/22
9		Documentación sobre arquitecturas de CPD	5 días	mié 12/10/22
10		Documentación sobre Cloud y virtualización	5 días	mié 19/10/22
11		Documentación sobre SDN	5 días	mié 26/10/22
12		Análisis de tecnologías y protocolos	5 días	mié 02/11/22
13		Análisis de controladores SDN	5 días	jue 03/11/22
14		Documentación sobre NFV	5 días	mié 09/11/22
15		Análisis sobre software a utilizar	3 días	mié 16/11/22
16		Entrega PEC 2	1 día	vie 11/11/22
17		**PEC 3**	**64 días**	**mié 28/09/22**
18		Creación de Laboratorio 1	10 días	sáb 12/11/22
19		Implementación de controlador SDN en ONOS	10 días	sáb 12/11/22
20		Resultados	2 días	vie 25/11/22
21		Creación de Laboratorio 2	10 días	mar 29/11/22
22		Implementación de tecnologías en OpenVswitch	10 días	mar 29/11/22
23		Resultados	1 día	mar 13/12/22
24		Valoración Productiva	1 día	mar 13/12/22
25		Entrega PEC 3	1 día	mar 13/12/22
26		**ENTREGA FINAL**	**79 días**	**mié 28/09/22**
27		Redacción de la memoria final	72 días	mié 28/09/22
28		Revisión u corrección de la memoria final	13 días	mar 20/12/22
29		Power Point	11 días	jue 22/12/22
30		Video Presentación	9 días	lun 26/12/22
31		Informe de autoevaluación	9 días	mar 27/12/22
32		**Final del TFG**	**1 día**	**dom 15/01/23**



Tareas críticas		Progreso de tarea		Línea base		Resumen		Tarea inactiva	
División crítica		Tarea manual		División de la línea base		Resumen manual		Hito inactivo	
Progreso de tarea crítica		solo el comienzo		Hito de línea base		Resumen del proyecto		Resumen inactivo	
Tarea		solo fin		Hito		Tareas externas		Fecha límite	
División		solo duración		Progreso del resumen		Hito externo			