Universidad de Valladolid

Escuela de Ingeniería Informática

Trabajo Fin de Grado

Grado en Ingeniería Informática

Analisis e Integración de los Componentes que Conforman una red SDN-Edge Computing

Autor:
D. José Ignacio Hernández Velasco

Tutores:
Dr. D. Jesús M. Vegas Hernández
D. Daniel Velasco Benito

Co-Tutor:
Dra. Dñ. Noemí Merayo Álvarez

Resumen

El presente trabajo pretende abordar el análisis e integración de los distintos componetes que conforman una red de acceso SDN-EdgeComputing. El objetivo es explicar los nuevos entornos de red y la necesidad de estos en el mundo actual ante el gran volumen de datos obtenidos por la aparición masiva de modernos dispositivos. Se explicará el estado actual de los distintos controladores SDN y las disparidades principales entre ellos, cómo opera la red, así como el software necesario para su funcionamiento. De igual manera, se analizarán las disimilitudes para la integración de 3 OLTs de diferentes fabricantes y qué supone esto en cuanto al desarrollo del software para esa red. Por último, se analizará el desarrollo de una herramienta para la monitorización y exportación de datos de la red, así como las métricas utilizadas para el análisis de la misma.

Abstract

The present work aims to address the analysis and integration of the different components that make up an SDN-EdgeComputing access network. The objective is to explain the new network environments and the need for these in today's world before the large volume of data obtained by the massive appearance of modern devices. It will explain the current status of the different SDN controllers and the main disparities between them, how the network operates, as well as the software necessary for its operation. Likewise, the dissimilarities for the integration of 3 OLTs from different manufacturers will be analyzed and what this means in terms of software development for that network. Finally, we will analyze the development of a tool for monitoring and exporting data from the network, as well as the metrics used to analyze it

Tabla de Contenidos

Lista de Figuras

Capítulo 1

Introducción

El presente TFG pretende abordar de una forma concisa y rigurosa los componentes que forman una red SDN-Edge Computing, así como las ventajas inherentes a esta tecnología.

Actualmente, nos encontramos con redes tradicionales que plantean problemas de escalabilidad y flexibilidad.

Con el gran flujo de datos con el que tratamos hoy en día en necesario una red que soporte y gestione este tráfico de la mejor forma posible.

El diseño y uso de software en los entornos de red está fuertemente vinculado al hardware que estemos usando, ya que cada fabricante usa un protocolo propietario diferente. Por lo tanto la elección de software es un punto importante en el diseño de la red.

La monitorización es otro de los problemas que se nos presenta, ya que este se ha empezado a tener en cuenta recientemente.

1.1. Objetivos

Estos son algunos de los problemas que presentan las redes actuales y por tanto se plantean una serie de obtjetivos para solucionarlos de la manera más eficiente:

- Presentar un solución de red SDN-Edge Computing: Con esta solución se pretende porporcionar una solución a los problemas que presentan las redes actuales, explicar en que consisten estas nuevas redes y por qué son una alternativa ante los problemas planteados.

- Abordar los elementos que componen una red de estas características: Se explicarán los componentes (software y hardware) que forman en su conjunto este tipo de redes.

- Resolver los problemas de software integrando software genérico e independiente de fabricantes: Se analizará los problemas en cuanto a la elección del software y se procurará realizar una integración de un software compatible con hardware de cualquier fabricante.

- Probar la interacción entre los componentes que conforman la red: Después de explicar cada uno de los componentes se explicará como se comunican entre sí y se realizará una prueba para comprobar su correcto funcionamiento.
- Establecer un sistema de monitorización, desarrollando y haciendo uso de las herramientas existentes: Se explicará el estado de la monitorización actual y se desarrollará un sistema de monitorización con las herramientas actuales.

1.2. Contexto actual

Hoy en día las redes actuales de las operadores se enfrentan a 3 problemas; Escalabilidad, Flexibilidad e Innovación

- Escalabilidad: La capacidad de crecimiento está limitada por el uso de nodos que son "cajas negras" y que requieren inversiones significativas para aumentar su capacidad o que deben ser sustituidas por nuevos elementos de mayor capacidad.
- Flexibilidad: El cambio de un nodo por otro se ve afectada por el dispositivo y el fabricante de ese dispositivo
- Innovación: Resulta costoso ofrecer nuevos servicios, ya el tiempo de desarrollo y la complejidad que requiere integrarlos en la red es muy elevada.

Se puede decir que las redes no han evolucionado a la par que los sistemas. La solución a resolver estos problemas es la virtualización de la red, la cual está compuesta compuesta por dos tecnologías fundamentales; NFV (virtualización de funciones de red) y SDN (redes definidas por software). A estas dos tecnologías le sumaremos una más, Edge Computing.

Debido al gran aumento de los dispositivos conectados a una red con capacidad de generar y transmitir datos, necesitamos de una comunicación efectiva que soporte ese tráfico. Teniendo esto en mente, surge el concepto de Edge Computing [5].

Hasta la fecha, el papel que Edge Computing ha venido desempeñando es insertar, filtrar, enviar y almacenar datos en los sistemas de la nube. Sin embargo, la tendencia de conectar a la red grandes cantidades de dispositivos *smart* (teléfonos, neveras, cafeteras, etc.) requieren de una arquitectura de cómputo con menor tiempo de respuesta para consultar y actuar sobre los datos.

Para ello, el uso de las redes SDN es la mejor forma de abordar la complejidad del problema con el que estamos tratando. Otro de los problemas que se plantean aquí es la elección de fabricantes para la creación de dichas redes. El diseño e integración de la red que se va a montar solventaría el problema del volumen de tráfico de datos y restaría importancia a la elección de fabricantes, intentando crear una solución genérica que funcionaría con cualquier hardware

1.3. NFV

La Virtualización de las Funciones de Red (NFV) [17] es un enfoque de red en evolución que permite la sustitución de dispositivos de hardware dedicados y costosos

tales como routers, firewalls y equilibradores de carga con dispositivos de red basados en software que se ejecutan como máquinas virtuales en servidores estándares de la industria.

NFV desacopla las funciones de la red de dispositivos de hardware dedicados y las traslada a servidores virtuales, y así se consolidan múltiples funciones en un único servidor físico. Este enfoque reduce los costos y minimiza la necesidad de envío de servicio técnico y el mantenimiento práctico, debido a que los dispositivos virtuales reemplazan dispositivos de red basados en hardware dedicados. En la Figura 1.1 está representado el paso de las funciones actuales a NFV.

La NFV tiene una serie de ventajas importantes, entre las que se incluyen:

- Menos espacio necesario para el hardware de red
- Menor consumo de energía de la red
- Menor costo de mantenimiento de la red
- Actualizaciones de red más sencillas y rápidas

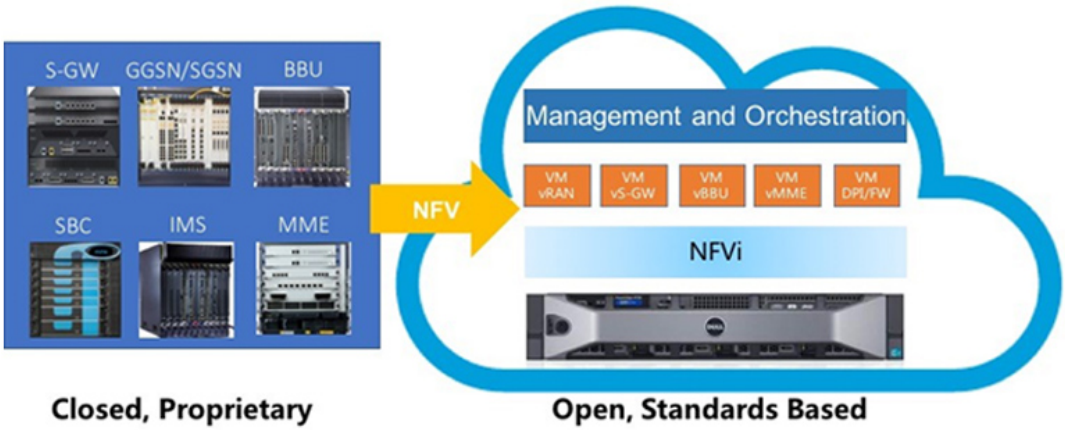

Figura 1.1: Esquema NFV

1.4. Edge Computing

El origen del término Cloud Computing [16] (o nube) no está claramente identificado. No obstante, las primeras referencias que se tienen datan de los años 90 cuando las empresas de telecomunicaciones, que antes ofrecían circuitos de datos punto a punto, empezaron a gestionar otros servicios de red privada virtual (VPN) a un costo mucho menor. Así surgió la nube.

En agosto de 2006 Amazon Web Services (AWS) [16] introdujo su Elastic Compute Cloud, considerado el primer servicio público de Cloud Computing disponible. Microsoft se anunció como "Azure" en octubre de 2008 y fue lanzado el 1 de febrero

de 2010 como Windows Azure, antes de ser rebautizado como Microsoft Azure el 25 de marzo de 2014.

Con la fuerte irrupción de las IoT (Internet of Things), grandes cantidades de datos digitales comenzaron a acumularse en los servidores de las empresas fabricantes (desde Amazon a Samsung, pasando por Telefónica, LG, SmartThings, wareables como Fitbit, etc). El negocio crece sin pausa. Según Cisco [3], en 2020 habrá unos 50.000 millones de dispositivos IoT. La clave está en los datos que esos sensores generan a cada segundo; cada vez más datos para gestionar, analizar, predecir, etc.

Todos esos volúmenes de datos resultan pasivos si no tienen interconexión. Podemos hacer muy inteligente un electrodoméstico, pero si este no se conecta, toda su capacidad quedará limitada.

El Edge Computing (de ahora en adelante, EC) es una arquitectura IT diseñada para acercar las aplicaciones y los datos a los usuarios o dispositivos que los necesiten. Mientras que el Cloud Computing impulsó la creación de unos cuantos mega Data Centers, el EC origina un entorno IT distribuido con un gran número de Micro Data Centers.

Las empresas están utilizando recursos IT basados en la nube: desde capacidad de computación hasta almacenamiento y aplicaciones. Para cumplir con las expectativas del usuario, necesitan complementar estos recursos basados en la nube. Aquí es donde entra en juego el EC. Los Edge Data Centers ayudan a lograr un mayor ancho de banda y una latencia más baja así como el favorecimiento del cumplimiento normativo respecto a la localización y a la privacidad de datos.

El objetivo del EC es ofrecer servicios con necesidades de baja latencia y altas velocidades. Sabiendo esto podemos decir que el EC es aplicable a multiples areas (Realidad Virtual, Gaming, IoT,etc)

1.5. SDN y Características de las SDN

Según Ciena [18], empresa de servicios de telecomunicaciones, define las redes definidas por software (SDN) como un enfoque arquitectónico que permite a la red ser controlada de manera inteligente o "programada", utilizando aplicaciones software. .

SDN permite la programación del comportamiento de la red de una manera controlada mediante aplicaciones de software que utilizan API abiertas. Con la apertura de las plataformas de red, generalmente cerradas, y la implementación de una capa de control de SDN común, los administradores pueden gestionar toda la red y sus dispositivos de forma consistente, independientemente de la complejidad de la tecnología de red subyacente.

Los puntos que caracterizan a las redes SDN son los siguientes:

- Programabilidad de la red: SDN permite que el comportamiento de la red se controle mediante el software que reside más allá de los dispositivos de red que proporcionan conectividad física. Como resultado, los operadores de red pueden adaptar el comportamiento de sus redes para soportar nuevos servicios e incluso clientes individuales. Al desacoplar el hardware del software, los operadores pueden introducir nuevos servicios diferenciados, libres de las limitaciones de las plataformas cerradas y patentadas.

- Centralizar inteligencia: SDN se basa en topologías de red centralizadas lógicamente que permiten la administración y el control inteligente de los recursos de la red. Se distribuyen métodos de control de red tradicional. Los dispositivos funcionan de forma autónoma con un conocimiento limitado del estado de la red. Con el tipo de control centralizado que ofrece una red basada en SDN, la administración, la restauración, la seguridad y las políticas de ancho de banda se encuentran muy optimizados; y una organización obtiene una visión integral de la red.

- Abstracción de la red: los servicios y las aplicaciones que se ejecutan en la tecnología SDN se abstraen de las tecnologías subyacentes y del hardware que proporcionan conectividad física de control de la red. Las aplicaciones van a interactuar con la red a través de las API, en lugar de las interfaces de administración estrechamente acopladas al hardware.

- Apertura: las arquitecturas de SDN dan comienzo a una nueva era de apertura que permite la interoperabilidad de proveedores múltiples, así como la promoción de un ecosistema no vinculado a proveedores. Las API abiertas soportan una amplia gama de aplicaciones, incluidas la orquestación de la nube, OSS/BSS(Sistema de soporte a las operaciones), SaaS (Software como servicio) y aplicaciones en red críticas para el negocio. Además, el software inteligente puede controlar el hardware de múltiples proveedores con interfaces de programación abiertas como OpenFlow (Protocolo de red que se explicará en el siguiente capítulo). Por último, desde el interior de SDN, los servicios de red y las aplicaciones inteligentes pueden ejecutarse dentro de un entorno de software común.

Una ventaja clave de la tecnología de SDN es la capacidad para que los operadores de red escriban programas que utilizan las API de SDN y dan a las aplicaciones el control del comportamiento de la red. SDN permite a los usuarios desarrollar aplicaciones orientadas a la red, monitorizar las condiciones de red de forma inteligente y adaptar automáticamente la configuración de la red, según sea necesario.

Capítulo 2

Controladores SDN

La arquitectura básica de las SDN [3] está compuesta por 3 capas: Infraestructura, Control y Aplicaciones. La lógica de la red en las SDN se encuentra en los controladores que mantienen la visión global de la red.

Los switch delegan su "inteligencia" al controlador, y su cometido pasa a ser el de conmutar paquetes. Es en el controlador donde se configurará el enrutamiento, el cual se encuentra abierto a la implementación de nuevas funcionalidades a través de sus API.

En cuanto a la capa de Infraestructura, está formada por los dispositivos de red. Estos exponen sus capacidades a través de una interfaz de control que permite la comunicación entre el controlador SDN y los dispositivos de red. Seguidamente, la capa de Control genera una interfaz que permite programar el control de las operaciones de reenvío por parte del propio controlador.

La interfaz de control se formaliza a través del protocolo OpenFlow, el cual se ha convertido en el protocolo *de facto* a utilizar para la conexión remota entre el controlador SDN y los switches. Por lo tanto, la elección del hardware y del software deben soportar el estándar OpenFlow.

En la capa de control se encuentra el controlador SDN, encargado de gestionar la conmutación del tráfico. El controlador equivaldría al sistema operativo de la red que maneja la comunicación entre las aplicaciones y los dispositivos.

Por último, la capa de Aplicación permite comunicar al controlador (mediante la APIs) tanto necesidades como el comportamiento que desean de la red, debido a que la definición de aplicaciones SDN es muy amplia (cubriendo desde servicios de redes, aplicaciones de negocio, etc.). Las diversas aplicaciones SDN hablan con la red de maneras diferentes y por lo tanto, requieren de interfaces diferentes. En la figura 2.1 se representan las 3 capas de la arquitectura SDN.

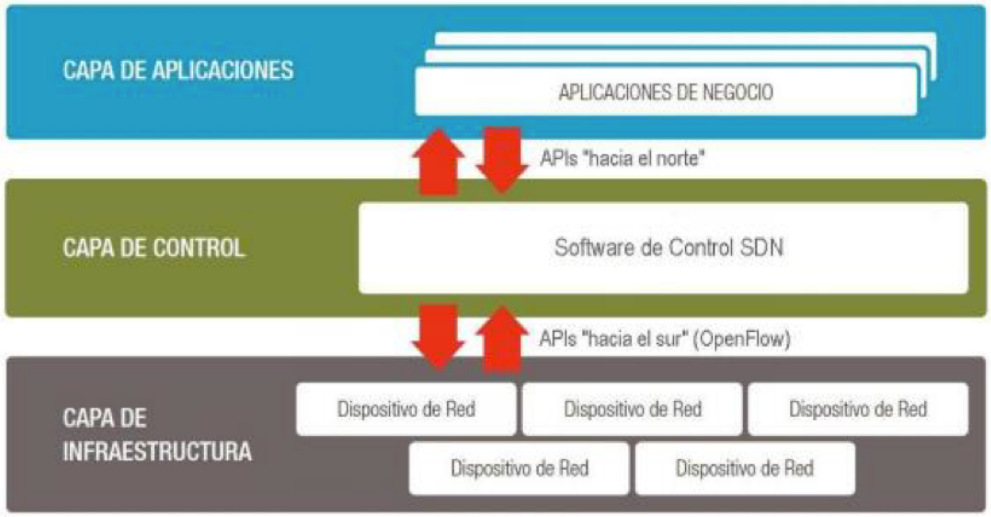

Figura 2.1: Arquitectura de Controlador SDN. Figura en [5]

2.1. OpenFlow

OpenFlow [11] es un protocolo que permite comunicar a los switches de red dónde enviar los paquetes requeridos. En una red convencional, cada switch tiene un software propietario que orienta las actividades a realizar. Con OpenFlow se estandarizan las decisiones de migración de paquetes, de modo que la red se puede programar independientemente de los switches individuales.

En un conmutador convencional, la transferencia de paquetes (la trayectoria de datos) y el enrutamiento de alto nivel (la trayectoria de control) suceden en el mismo dispositivo. Un switch separa la trayectoria de datos de la trayectoria de control. La porción de trayectoria de datos reside en el propio conmutador y un controlador separado toma las decisiones de enrutamiento de alto nivel. El conmutador y el controlador se comunican por medio del protocolo OpenFlow.

Muchas empresas establecidas, incluyendo IBM, Google y HP, han utilizado completamente o han anunciado su intención de soportar el estándar de OpenFlow. Big Switch Networks [11], una compañía de SDN con sede en Palo Alto, California, ha implementado redes de OpenFlow que funcionan sobre redes tradicionales, lo que hace posible colocar máquinas virtuales en cualquier centro de datos para obtener capacidad de computación. A principios de 2012, la red interna de Google estaba soportada completamente con OpenFlow. En la Figura 2.2 se muestra el funcionamiento de OpenFlow

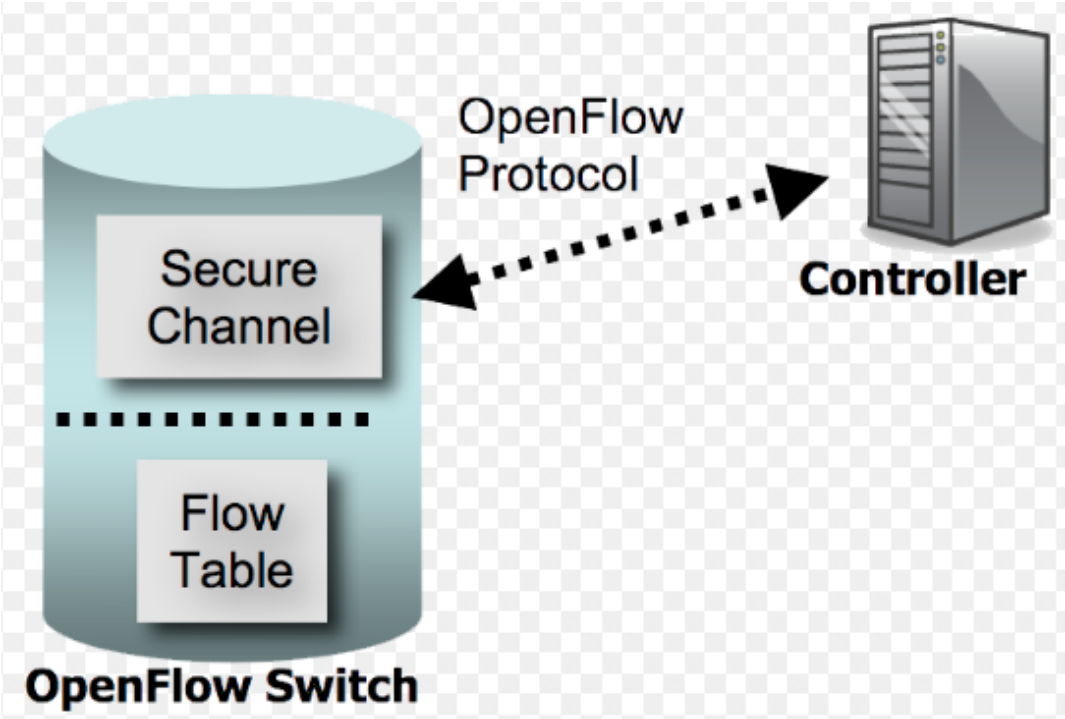

Figura 2.2: Funcionamiento de OpenFlow

2.2. Controladores Openflow y Factores para elegir un Controlador

Un controlador Openflow [3] ofrece una interfaz de programación para los conmutadores OpenFlow. Un controlador SDN puede ser descrito como un sistema o colección de sistemas que ofrecen:

- Gestión del estado de la red: implica una base de datos. Dichas bases sirven como un repositorio para la información de los elementos de red gestionados, incluyendo el estado de la red, alguna información de configuración temporal e información sobre la topología de la red.
- Un modelo de datos de alto nivel que captura las relaciones entre los recursos gestionados, las políticas y otros servicios prestados por el controlador. En muchos casos estos modelos de datos se construyen utilizando el lenguaje de modelado Yang.
- Un mecanismo de descubrimiento de dispositivos, topología y servicio; un sistema de cálculo de ruta y, potencialmente, otros servicios de información centrados en la red o en los recursos.
- Una sesión de control segura sobre el Protocolo de Control de Trasmisión (TCP por su nombre en ingles *Transmission Control Protocolo* entre el controlador y los agentes asociados en los elementos de la red, por ejemplo con el uso del protocolo TLS.

- Un protocolo basado en estándares (OpenFlow) para obtener el estado de la red impulsado por las aplicaciones de los elementos de red.

- Un conjunto de APIs, a menudo RESTful que exponen los servicios del controlador a las aplicaciones de gestión. Esto facilita la mayor parte de la interacción del controlador con estas aplicaciones.

Esta interfaz se representa a partir del modelo de datos que describe los servicios y funciones del controlador. En algunos casos, el controlador y su API son parte de un entorno de desarrollo que genera el código de la API a partir del modelo de datos.

Algunos controladores SDN que soportan OpenFlow: ODL, Ryu, Trema, POX, ONOS.

Los factores [3] a tener en cuenta cuando elegimos un controlador SDN son los siguientes:

- Soporte OpenFlow: al elegir un controlador los administradores de red necesitan conocer las características de las versiones de OpenFlow que el controlador soporta, así como las posibilidades que ofrece el proveedor para migrar a las nuevas versiones del protocolo, tales como la v1.3 y la v1.4. Una razón por la que esto es necesario, es que algunas funciones importantes, como por ejemplo el soporte de IPv6, no son parte de OpenFlow v1.0 pues se incluyen a partir del estándar OpenFlow v1.2.

- Virtualización de red: debido a los beneficios que ofrece la virtualización de red, un controlador SDN debe soportarla. Esta característica permite a los administradores crear dinámicamente las redes virtuales basadas en políticas, disociadas de las redes físicas, para satisfacer una amplia gama de requisitos, como por ejemplo, la ampliación horizontal de la capacidad, sin afectar a los flujos existentes.

  Otra de las muchas ventajas de la virtualización de red, es que permite un completo aislamiento entre cada segmento de red, lo que es muy útil por razones de seguridad. Por ejemplo, para mantener aislados los datos generados por un grupo de usuarios de otros usuarios y permitir a los desarrolladores de aplicaciones ejecutar las mismas en un entorno de trabajo sin afectar el tráfico.

  Para cumplir estos requisitos de manera eficiente, en los controladores SDN se deben configurar las redes virtuales de forma centralizada, con total aislamiento unas de otras. De igual manera, dichas configuraciones deben estar automatizadas.

- Funcionalidad de la red: para lograr mayor flexibilidad en términos de cómo los flujos son enrutados, es importante que el controlador SDN pueda tomar decisiones de enrutamiento basado en múltiples campos de la cabecera de OpenFlow. También es importante que el controlador pueda definir los parámetros de QoS flujo por flujo.

  Otra importante funcionalidad en un controlador SDN es su capacidad para descubrir múltiples caminos desde el origen del flujo a su destino y para dividir el tráfico de un flujo dado a través de múltiples enlaces. Esta capacidad elimina

la necesidad de STP (Protocolo de Árbol Extendido o *Spanning Tree Protocol*) aumentando el rendimiento y la escalabilidad de la red permitiendo así mismo, eliminar el requisito de añadir a la complejidad de la red nuevos protocolos como TRILL (*Transparent Interconnection of Lots of Links*) o SPB (*Shortest Path Bridging*).

- Escalabilidad: una consideración fundamental en este apartado es el número de conmutadores o switches que un controlador SDN puede soportar. En la actualidad, se debe esperar que los controladores soporten un mínimo de 100 switches, pero en última instancia esto depender de las aplicaciones que soportan.

  Otro factor que limita la escalabilidad de una red SDN es la proliferación de entradas en la tabla de flujo. Al evaluar los controladores SDN, es necesario asegurarse que el controlador puede disminuir el impacto de sobrecarga de difusión de red, la cual limita la escalabilidad de la arquitectura de red implementada y reduce al mínimo la proliferación de las entradas de la tabla de flujo.

  Otro aspecto de la escalabilidad es la capacidad del controlador de SDN para crear una SDN que pueda abarcar múltiples sitios. Esta habilidad permite el movimiento de máquinas virtuales y el almacenamiento virtual entre sitios. Para maximizar el beneficio de esta capacidad, el controlador SDN debe permitir que las políticas de red para el enrutamiento y reenvío se apliquen automáticamente para la migración de servidores y/o almacenamiento.

- Rendimiento: una de las principales funciones de un controlador SDN es establecer flujos. Por ello, dos de los indicadores claves de rendimiento asociados con un controlador SDN son el tiempo de conformación de flujo y el número de flujos por segundo que puede establecer el controlador. Estas métricas de desempeño influyen en gran medida cuando se requiere añadir controladores, como por ejemplo, cuando los switches inician más flujos de los que pueden ser soportados por el controlador o los controladores SDN existentes.

- Programación de red: una de las características fundamentales de las SDN es la existencia de interfaces para la programación del controlador, lo que posibilita que se ofrezcan varias funcionalidades. Algunos ejemplos de programación que se deben buscar en un controlador SDN, son la capacidad de redirigir el tráfico (por razones de seguridad se puede disponer que el tráfico entrante a un servidor pase a través de un corta fuegos o firewall, pero para no consumir los recursos del servidor de seguridad con tráfico limpio se puede decidir que no pase por el firewall el tráfico saliente del mismo servidor) y la posibilidad de aplicar filtros sofisticados a los paquetes los cuales pueden ser pensados como ACLs (Listas de Control de Acceso o *Access Control List* por sus siglas en inglés) dinámicas e inteligentes como combinaciones complejas de múltiples campos de cabecera de paquetes.

- Soporte comercial al controlador SDN: dado el creciente interés en las SDN, numerosos fabricantes han entrado en el mercado y muchos más han anunciado su intención de hacerlo. Debido a la volatilidad del mercado SDN en general, y del mercado del controlador SDN en particular, las organizaciones que deben

evaluar controladores SDN para sus redes deben centrarse no sólo en los atributos técnicos del controlador, sino también en las características del vendedor. Entre estas, se encuentra la competencia técnica y financiera del proveedor pues las organizaciones no pueden permitirse tener las SDN perjudicadas por adquirir controladores de un proveedor que no puede mantenerse al día en el cambiante entorno SDN.

Capítulo 3

Contexto de la red

En este capítulo se pretende explicar los diferentes componentes que conforman la red que se va a implementar. Se comenzará exponiendo los componentes que conforman una red PON; se continuará con los componentes software que estamos utilizando (ONOS y VOLTHA) y por último, se formulará un esquema completo de la red y la topología que se está usando *in situ*.

La red con la que estamos trabajando está formada por 6 switches que hablan OpenFlow, un conjunto de Host en los que se almacenan los diferentes servicios y componentes software, una OLT y una ONU. La creación y gestión de las diferentes MV que almacenan los servicios se hace a través de OpenNebula, una plataforma de cloud computing.

3.1. Redes PON

A lo largo del tiempo, las tecnologías usadas en las redes de acceso se han ido actualizando. Se comenzó con redes implementadas solo con cobre y posteriormente se pasó a redes híbridas, implicando un aumento del ancho de banda y disminución de latencia. En los últimos años, las compañías de telecomunicaciones de todo el mundo han empezado a modernizar su infraestructura adoptando la tecnología, denominada fibra, hasta el hogar (FTTH). Esto a hecho que los proveedores de estos sistemas estén avanzando rápidamente. Existen dos tipos importantes de sistemas que hacen posibles las conexiones de banda ancha FTTH: redes ópticas activas (AON) y redes ópticas pasivas (PON).

Una red óptica pasiva (PON) [15] es un sistema de red con cableado de fibra óptica que envía la señal de todo, o casi todo, el recorrido hasta el usuario final. El sistema se describe de diferente forma según dónde termine la red PON, así pues, tendríamos: fibra hasta la acera (FTTC), fibra hasta el edificio (FTTB) o fibra al hogar (FTTH). Los componentes principales de PON son los siguientes: OLT, ONT/ONU.

3.1.1. Componentes de una Red PON

Una PON (Red Óptica Pasiva) es un terminal de línea óptica (OLT) [15], situada en la centralita de la empresa de comunicaciones proveedora, y en varias unidades de red óptica (ONU), ubicadas cerca de los usuarios finales. Actualmente hay dos

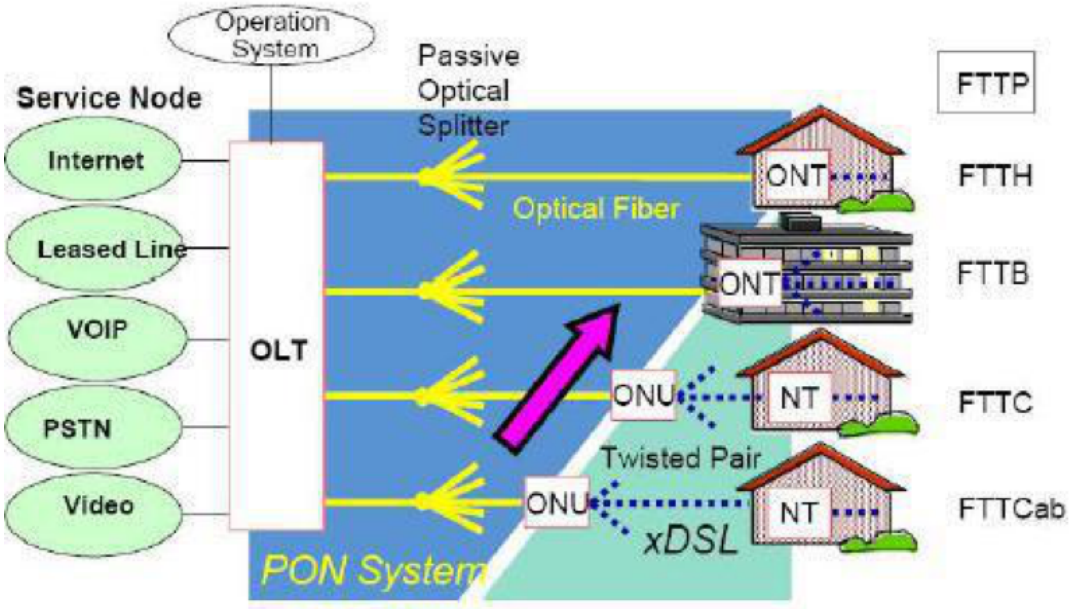

Figura 3.1: Esquema Redes PON. Obtenido de [15]

estándares principales de PON: Red óptica pasiva con capacidad de Gigabit (GPON) y Red óptica pasiva preparada para Ethernet (GEPON). Las red PON que está integrada en España son las GPON. La Figura 3.1 muestra el sistema de red PON

Un sistema de red óptica pasiva Gigabit (GPON) generalmente se compone de un terminal de línea óptica (OLT) en la centralita del proveedor del servicio y varias unidades de red óptica (ONU) o terminales de red óptica cerca de los usuarios finales. Se utiliza la red de distribución óptica (ODN) durante la transmisión entre OLT y ONU/ONT. La ODN es el medio de transmisión óptica para la conexión física de las ONU a las OLT. En la figura 3.1 la ODN hace referencia al tramo de la conexión entre la OLT y las distintas ONU.

3.1.2. Terminal de línea óptico (OLT)

OLT es un equipo que integra la función de switch L2/L3 en el sistema GEPON [15]. En general, el equipo OLT contiene un bastidor, un módulo de control de conmutación, un ELM (módulo de enlace EPON, tarjeta PON), un sistema de protección de redundancia, módulos de fuente de alimentación y ventiladores. En estas partes, la tarjeta PON y la fuente de alimentación admiten el intercambio en caliente. La función principal del OLT es controlar desde una centralita la información transmitida en ambas direcciones a través de la ODN . La distancia máxima admitida de transmisión a través de la ODN es de 20 km. La OLT controla los dos sentidos de la transmisión de información: ascendente (obteniendo una clase diferente de distribución del tráfico de información y voz de los usuarios) y descendente (obteniendo tráfico de datos, voz y vídeo desde una red metro o una red de larga distancia y enviando todos los módulos ONT en el ODN).

Es importante aclarar que las diferencias entre las OLT dependen de los fabricantes y aunque el funcionamiento es igual desde el punto de vista del usuario, desde la perspectiva de un desarrollador surge un problema: cada OLT funciona internamente con un protocolo propietario diferente. Es decir, con cada hardware que adquiramos deberemos estar en contacto con dicho proveedor para saber de sus correspondientes especificaciones y manejo de su OLT.

Esto genera una problemática desde el abordaje que mencionábamos previamente. Apuntan las redes definidas por software, cuya ventaja reside en controladores centrales independientes al hardware que usemos. Es decir, tenemos una OLT que habla un lenguaje específico (protocolo propietario) y una red SDN que habla otro lenguaje distinto (OpenFlow). Surge pues una necesidad de abstraer la capa propietaria.

Para esta SDN usaremos varias OLT. Partiendo de una integración de una OLT GPON de Celéstica, la siguiente integración será con una OLT de Tibit XGSPON (XGSPON;10 Gigabit symetric PON). Por último, se realizará una integración con una OLT XGSPON de Edgecore.

La integración con esta última OLT resulta diferente en cuanto a las otras dos, lo cual se mostrará más adelante.

Tanto GPON como XGSPON son estándares en la transmisión de datos de las redes PON. La principal diferencia entre GPON y XGSPON es la tasa de transferencia de datos, teniendo GPON una tasa de transferencia de subida de 1.2G y de 2.5G en bajada y XGSPON una tasa de 10G simétrica. Las Figuras 3.2 y 3.3 representan las OTLs que se van a integrar.

Figura 3.2: OLT de Tibit. Obtenido de [22]

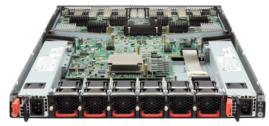

Figura 3.3: OLT de Edgecore. Obtenido de [19]

3.1.3. Unidad de Red Óptica (ONU) y Terminal de red Óptico (ONT)

La ONU convierte las señales ópticas transmitidas a través de la fibra en señales eléctricas para su posterior distribución a los suscriptores individuales. En general,

existe cierta distancia, u otra red de acceso, entre la ONU y las instalaciones donde se encuentra usuario final. Además, la ONU puede enviar, agregar y multiplexar diferentes tipos de datos provenientes del cliente y enviarlos en sentido ascendente a la OLT. Grooming es un proceso de gestión de la ONU que optimiza y reorganiza el flujo de datos para que estos sean transportados más eficazmente. La OLT admite la asignación de ancho de banda para permitir así una entrega de datos fluida y sin problemas, ya que esta generalmente se multiplexa en tiempo. Se puede conectar la ONU con la red de casa del cliente mediante varios métodos y tipos de cable, como por ejemplo el cable de par trenzado de cobre o el cable coaxial, con fibra óptica o con Wi-Fi.

La ONT es un término de la UIT-T, mientras que ONU es un término del IEEE. Ambos se refieren al equipo del usuario en el sistema GEPON. En la práctica, sin embargo, hay una pequeña diferencia entre ONT y ONU según su ubicación. En la figura 3.1 se aprecia que la ONT se localiza *in situ* en las instalaciones del cliente, sin embargo la ONU no se encuentra físicamente en las instalaciones del mismo.

3.2. VOLTHA

VOLTHA [21] es un proyecto de código abierto para crear una abstracción de hardware para equipos de acceso de banda ancha. Es compatible con el principio de múltiples proveedores.

Actualmente, VOLTHA proporciona un sistema común de control y gestión GPON, independiente del proveedor, para un conjunto de dispositivos de hardware PON de caja blanca y específicos del proveedor.

En su interfaz hacia el norte, VOLTHA abstrae la red PON para que aparezca como un conmutador Ethernet programable a un controlador SDN. En su lado sur, VOLTHA se comunica con dispositivos de hardware PON utilizando protocolos específicos del proveedor a través de adaptadores OLT y ONU. La figura 3.4 muestra la aruitectura de VOLTHA.

3.2.1. Definición de VOLTHA

Se puede decir que VOLTHA es un "traductor" que habla varios ïdiomas"; en la capa Norte se comunicaría con ONOS a través del protocolo OpenFlow, y en la capa Sur se comunicaría con las OLT y ONT por medio de los adaptadores. Es importante tener en cuenta que se precisa de un adaptador específico para una OLT determinada, es decir, dependiendo del fabricante será necesario usar un adaptador u otro debido a que cada OLT usa su propio protocolo propietario. Al ser VOLTHA un proyecto Open Source, se pueden consultar todos los adaptadores que ya tiene desarrollados en su repositorio de github. Es importante aclarar que los adaptadores se actualizan

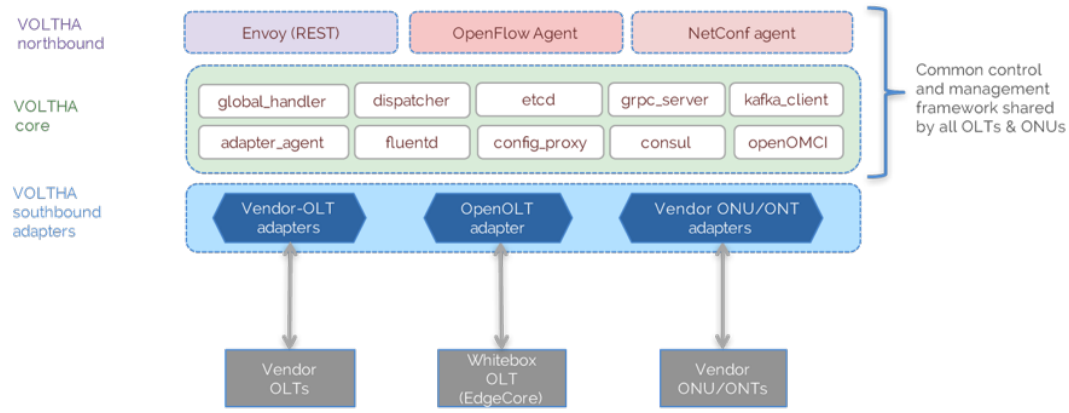

Figura 3.4: Arquitectura VOLTHA. Obtenido en [21]

conforme se van realizando nuevas pruebas de integración y encontrando fallos, así como a medida que se va actualizando la versión de VOLTHA. Para la integración de la red se está usando la versión de VOLTHA 1.6

3.2.2. Adaptadores VOLTHA

Cuando se ha explicado en qué consiste una OLT, se han listado las OLT que se emplearán para la integración de la red, lo que implica un uso de los adaptadores específicos de cada una, no siendo el caso en la OLT XGSPON de Edgecore, para el cual se intentará usar un adaptador genérico.

Este es el llamado adaptador OpenOLT, el cual podrá emplearse para comunicarnos con cualquier OLT. La principal ventaja de usar un adaptador genérico es que no tendremos que preocuparnos por el hardware que usemos. Tendremos un adaptador único válido para cualquier OLT que compremos, por lo que ya no estaríamos limitados por los fabricantes.

De la misma forma que tenemos adaptadores específicos para las OLT, también tenemos adaptadores concretos para las ONT; igual que tenemos un adaptador genérico para las OLT, también disponemos de un adaptador común para las ONT, el llamado brcm-openomci-onu, un adaptador único para toda ONT. Usaremos este adaptador para la integración de Tibit y EdgeCore y un adaptador específico para el caso de la OLT de Celéstica, en el que emplearemos una ONU diferente.

Los adaptadores son encargados de la gestión y control de sus OLT, la instalación de los flows que se instalan y las ONT que se añaden a las OLT. Estos son los principales componentes encargados de la comunicación de los mensajes hacia la OLT.

3.2.3. OpenOLT

Las OLTs a bajo nivel funcionan igual en prácticamente todos los casos. Sin embargo, lo que hace que sea difícil trabajar con cada una de ellas, es el uso de protocolos propietarios que limitan a la hora de desarrollar software. OpenOLT surge

como una forma de evitar esa dependencia. Sin embargo, aunque trabajemos con este adaptador genérico, las OLT siguen funcionando internamente con su propio protocolo.

Funcionamiento de OpenOLT

La principal razón por la que hoy día OpenOLT funciona con las OLT de Edgecore, es que actualmente son la única empresa que está desarrollando internamente sus OLT como un agente que traduce los mensajes de VOLTHA a su protocolo propietario. Estas son denominadas OLTs whitebox. Previsiblemente, esta práctica se irá extendiendo al resto de fabricantes conforme avance el tiempo. El proyecto CORD, principal impulsor de esta tecnología, cuenta con el apoyo de muchas empresas.

Es importante aclarar que el funcionamiento no es óptimo, ya que día a día se realizan prubeas de los adaptadores, se sacan fallos o se desarrollan nuevas funcionalidades para los adaptadores. Es decir, la producción de todo esto se estima para largo plazo.

En la Figura 3.5, podemos ver cómo funcionaría OpenOLT junto con la OLT. Por un lado, tenemos a VOLTHA con el adaptador OpenOLT. Este se comunica con la OLT a través de una api gRPC con el driver interno de la OLT, siendo este driver el que traduce el mensaje al protocolo propietario de la OLT.

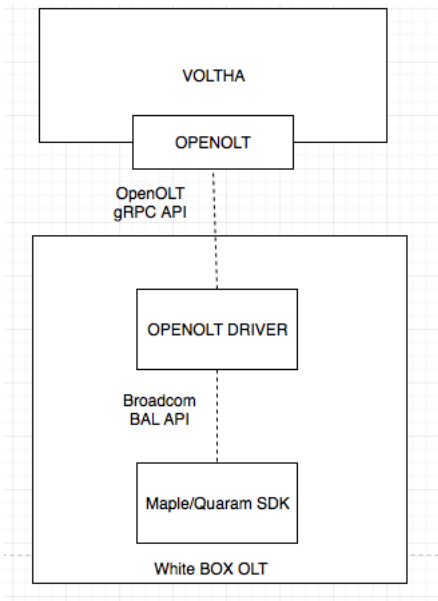

Figura 3.5: Funcionamiento OpenOLT.

OpenOLT utiliza el protocolo gRPC (*Google Remote Procedure Call*), desarrollado por Google para conectar microservicios. Las características son las siguientes [12] :

- Construido sobre HTTP/2
- Bajo consumo de CPU

- Serialización con Proto3 (Protocol Buffers). Método de serialización de datos estructurados
- Funciona sobre IDLs (*Interface Definition Languaje*). Ofrece la sintaxis para definir los procedimientos que queremos invocar de forma remota
- Protocolo bidireccional con control de flujo de datos y con control de capacidad para múltiples peticiones, sobre todo TPC.

El funcionamiento del protocolo es sencillo. En gRPC una aplicación cliente, como en nuestro caso OpenOLT, llama a métodos de la aplicación servidora de forma transparente. De esta forma es posible la comunicación entre clientes y servidores en diferentes lenguajes. El funcionamiento de gRPC se representa en la figura 3.6.

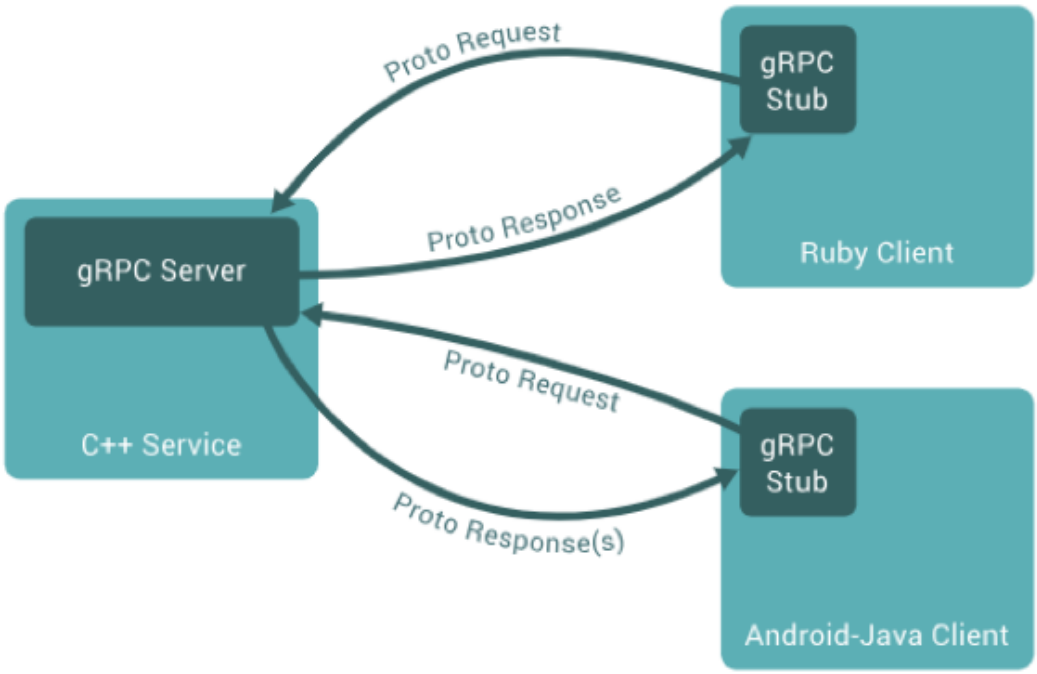

Figura 3.6: Funcionamiento protocolo gRPC

Se puede apreciar cómo se comunicaría nuestro adaptador con las diferentes OLTs.

Aunque como hemos dicho anteriormente, se precisa una OLT que internamente ya tenga un software que traduzca los mensajes a su protocolo propietario. Al mismo tiempo, el agente interno también debe usar gRPC para poder comunicarse con OpenOLT. La intención sería migrar todo a OpenOLT y no tener que usar adaptadores específicos. Hoy día, las capacidades para poder desarrollar esto son limitadas.

Ventajas de OpenOLT

Las ventajas que nos ofrece OpenOLT son muchas, siendo la más característica un mismo software que funciona a la vez con múltiples OLTs distintas y de distintos

fabricantes. Se cuenta con una reducción de complejidad enorme, ya que actualmente es necesario no solo trabajar con distintos adaptadores, si no tener en cuenta el funcionamiento interno de las distintas OLTs y los protocolos que usan para poder hacer cambios en esos adaptadores.

Con OpenOLT no es indispensable contar con todo lo anteriormente expuesto. El adaptador no es responsable del funcionamiento de la OLT con la que trabaja, en cambio, esta responsabilidad pasaría al propio agente interno de la OLT que traduce los mensajes a su protocolo propietario. Por otra parte, resulta más sencillo trabajar con un código único que con varios lo que abocaría a otra forma de reducir la complejidad.

Como ya se ha mencionado previamente, la primera empresa que está apostando por esta tecnología es Edgecore, impulsora del proyecto CORD. Las previsiones de futuro es que más empresas se sumen a esta iniciativa tecnológica. En las Figuras 3.7 y 3.8 se aprecia de una forma más clara sobre cuál es el propósito del adaptador OpenOLT, y cuál es la situación actual en VOLTHA. En la figura 3.7 tenemos un esquema de múltiples adaptadores para diversas OLT; cada adaptador se comunica con su OLT. En la figura 3.8 se presenta el esquema del adaptador OpenOLT, cuyo propósito es el de unificar la lógica de las OLT en un único adaptador funcional para cualquier proveedor.

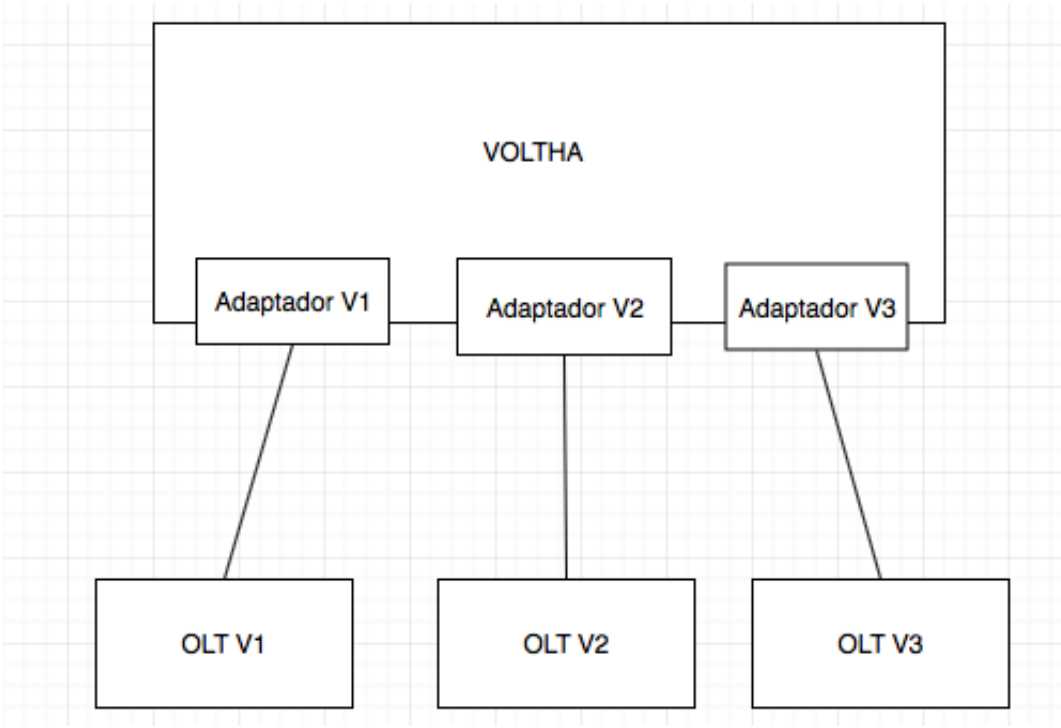

Figura 3.7: Esquema de varios Adaptadores

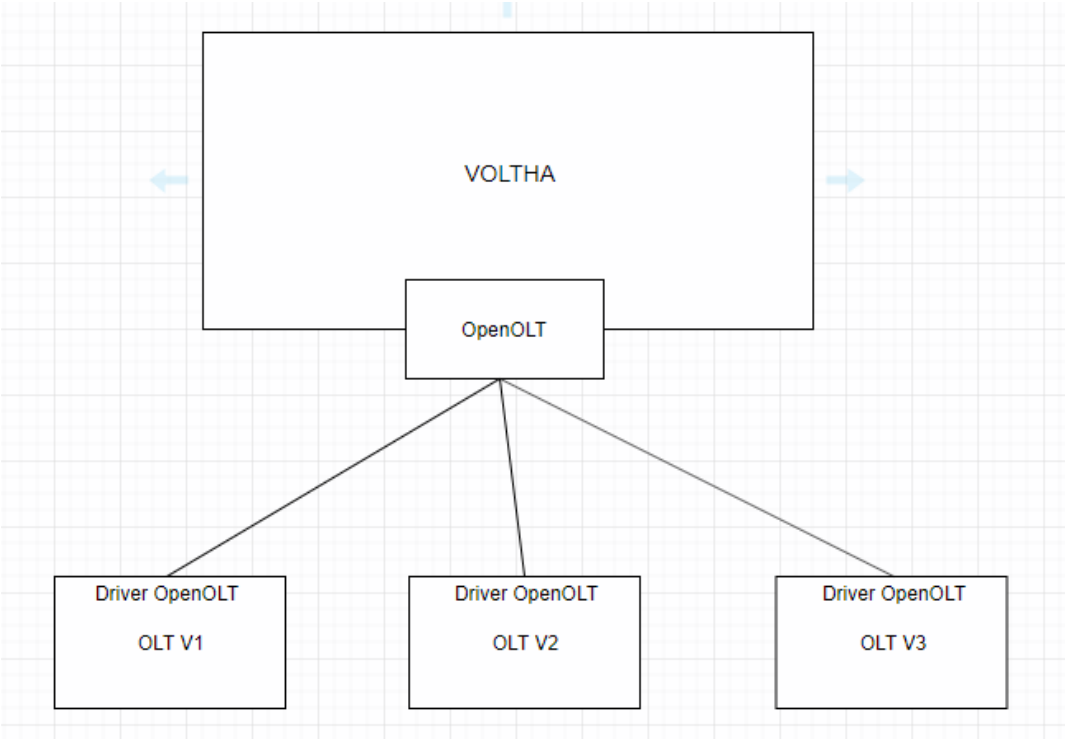

Figura 3.8: Esquema de único Adaptador

3.2.4. OpenOMCI

Previamente se ha comentado que VOLTHA hace uso de adaptadores para comunicarse con los diferentes dispositivos. El adaptador genérico del que hemos hablado es el OpenOLT, sin embargo, también disponemos de un adaptador común para comunicarnos con las diferentes ONT: el adaptador OpenOMCI.

OMCI

El protocolo OMCI [4] es el protocolo estándar de GPON para el control por parte de la OLT de las ONT. Este protocolo permite:

- Establecer y liberar conexiones en la ONT
- Gestionar los puertos físicos de la ONT
- Solicitar información de configuración y estadísticas de rendimiento
- Informar autónomamente al operador del sistema de eventos, tales como cortes de fibra

El protocolo OMCI se ejecuta sobre una conexión GEM (GPON Encapsulation Method) [4] entre la controladora de la OLT y la controladora de la ONT, establecido durante la fase de arranque de la ONT. El protocolo OMCI es asimétrico. El símil que se plantea en la explicación es el siguiente: el OLT es el maestro y la ONT es el

esclavo. Un único OLT, empleando diversas instancias del protocolo sobre canales de control independientes, puede controlar múltiples ONTs. Los requerimientos de la OMCI dados en la recomendación G.984.4 de la ITU-T son necesarios para manejar la ONT en las siguientes áreas:

- Gestión de la configuración
- Gestión de fallos
- Gestión del rendimiento
- Gestión de la seguridad

Funcionamiento de OpenOMCI

OMCI es el protocolo que hablan las diferentes ONT en GPON, por lo que el adaptador OpenOMCI manda una secuencia de mensajes de activación a la ONT.

El mecanismo de activación consta de 7 estados [2]:

- O1-Initial state
- O2-Standby state
- O3-Serial-Number state
- O4-Ranging state
- O5-Operation state
- O6-POPUP state
- O7-Emergency Stop state

Definición de los estados:

- O1-Initial state: la ONU se enciende. Se afirma LOS / LOF. Una vez que se recibe el tráfico en sentido descendente, se borran LOS y LOF y entonces la ONU pasa al estado de espera (O2).
- O2-Standby state: la ONU recibe el tráfico descendente. La ONU espera los parámetros de la red global. Una vez que se recibe el mensaje Overstream-Overhead, la ONU configura estos parámetros (por ejemplo, el valor del delimitador, el modo de nivel de potencia y el retraso preasignado). Seguidamente pasa al estado del número de serie (O3).
- O3-Serial-Number state: al responder a las solicitudes de número de serie enviadas por la OLT, la ONU se da a conocer a la OLT y permite que la OLT descubra el número de serie de la ONU. Una vez que la ONU ha respondido a una solicitud de número de serie, espera la asignación de ID de ONU única de la OLT. La ONU-ID se asigna utilizando el mensaje Assign-ONU-ID. Una vez asignada, la ONU pasa al estado de rango (O4).

La OLT puede, a su discreción, usar el mensaje Extended-Burst-Length para comunicar los parámetros de la sobrecarga extendida a todas las ONU en la PON. Si la ONU en estado de número de serie (O3) recibe el mensaje Extended-Burst-Length antes de recibir cualquier solicitud de número de serie, configura las longitudes de preámbulo de tipo 3 de acuerdo con los valores recibidos.

- O4-Ranging state: la transmisión en sentido ascendente desde las diferentes ONU debe sincronizarse con los límites de trama GTC en sentido ascendente. Para hacer que las ONU parezcan estar a la misma distancia de la OLT, se requiere un retardo de ecualización por ONU. Este retardo de ecualización se mide cuando la ONU está en el estado de rango. Una vez que la ONU recibe el mensaje Ranging-Time, pasa al estado de operación (O5).

- O5-Operation state: una vez en este estado, la ONU puede enviar datos en sentido ascendente y mensajes PLOAM como lo indica la OLT. Se pueden establecer conexiones adicionales con la ONU, según sea necesario, mientras se encuentre en este estado. Una vez que la red tiene rango, y todas las ONU funcionan con su retardo de ecualización correcto, todas las ráfagas en sentido ascendente se sincronizarán entre todas las ONU. Las transmisiones ascendentes llegarán por separado, cada una en su ubicación correcta dentro de la trama GTC ascendente.

- O6-POPUP state: a ONU ingresa a este estado desde el estado de Operación (O5) luego de la detección de las alarmas LOS o LOF. Al ingresar al estado POPUP (O6), la ONU detiene inmediatamente la transmisión en sentido ascendente. Como resultado, la OLT detectará una alarma LOS para esa ONU.

  En el estado POPUP, la ONU primero intenta volver a adquirir la señal óptica y restaurar la sincronización de trama GTC, eliminando así las condiciones LOS y LOF. Una vez que tiene éxito, la ONU comienza a procesar el campo PCBd de las tramas GTC en sentido descendente y reinicia la máquina de estado de sincronización de supertrama. Se tiene que tener en cuenta que en el caso de la protección de Tipo B, la señal puede provenir de la OLT de respaldo o de la OLT primaria.

  Mientras se encuentra en el estado POPUP, la ONU genera un evento de recepción de mensaje PLOAM solo en respuesta a los mensajes Disable-ONU-ID, Deactivate-Serial-Number y POPUP. Si la ONU recibe un mensaje POPUP dirigido, pasa al estado de operación (O5). Si la ONU recibe un mensaje POPUP de difusión, pasa al estado de rango (O4).

  Una vez que la ONU está en el estado de Operación (O5), la OLT puede probar la ONU antes de devolverla al servicio completo. En particular, es posible que se haya programado un evento de cambio de clave de cifrado en el estado POPUP (O6). Para garantizar una recuperación correcta en tal situación, la OLT debe reiniciar el intercambio de claves y el procedimiento de cambio con la ONU.

  Si la ONU no puede volver a adquirir la señal óptica o restaurar la sincronización de trama GTC, no recibirá el mensaje POPUP (emitido o dirigido) y pasará al estado inicial (O1), luego del tiempo de espera (TO2).

- O7-Emergency Stop state: una ONU que recibe un mensaje Disable-Serial-Number con la opción 'deshabilitar' pasa al estado de parada de emergencia (O7) y apaga su láser.

  Durante la parada de emergencia, la ONU tiene prohibido enviar datos en sentido ascendente. Si la ONU no se mueve al estado de parada de emergencia, es decir, después de que el mensaje Disable-Serial-Number se haya enviado tres veces, la OLT continúa recibiendo las transmisiones de la ONU en las asignaciones de ancho de banda ascendente proporcionadas. Una alarma DFi se afirma en la OLT.

  Cuando el funcionamiento defectuoso de la ONU desactivada es fijo, la OLT puede activar la ONU para que vuelva a funcionar correctamente. Dicha activación se logra enviando un mensaje Disable-Serial-Number con la opción 'habilitar' a la ONU. Como resultado, la ONU vuelve al estado de espera (O2). Todos los parámetros (incluidos el número de serie y la ONU-ID) se reexaminan.

En la Figura 3.9 podemos ver de una manera gráfica los diferentes estados y la transición entre ellos.

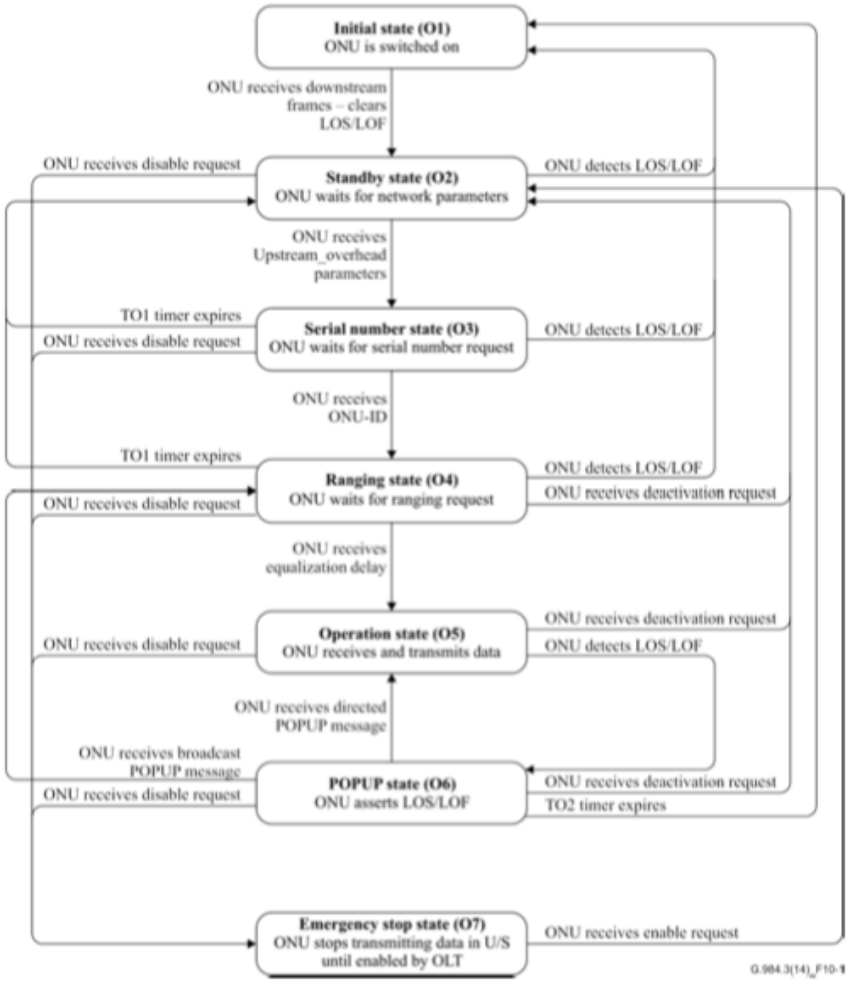

Figura 3.9: Estados OMCI. Obtenido en [1]

3.3. ONOS

En esta parte vamos a hablar sobre los servicios software que forman parte de la red, en este casode ONOS, el controlador elegido para esta red de acceso. Previamente hemos hablado de VOLTHA y la posibilidad que ofrece de comunicarnos con múltiples OLT y ONT. El uso de VOLTHA es necesario. Tenemos varias OLT que usan diferentes protocolos. El mismo problema ocurre en el caso de los switches. En esta red disponemos de 6 switches que se comunican a través de Openflow, por lo que necesitamos un controlador que también tenga la posibilidad de usarlo. Es aquí donde entra ONOS, el encargado de tomar el control de los switches a través del protocolo Openflow. ONOS no es el único controlador disponible, sin embargo es uno de los más usados actualmente. Otro de los controladores que podemos encontrar en esta línea es ODL (OpenDaylight)

3.3.1. Funcionamiento de ONOS

El núcleo de ONOS está diseñado con una arquitectura modular. Debido a que los proveedores de servicio requieren la capacidad de escalar sus redes, ONOS puede hacerlo para adaptarse a un sistema de dispositivos distribuidos físicamente. Esto permite agregar nuevos conmutadores o componentes sin interferir con el resto del sistema.

Mientras que el núcleo de ONOS se distribuye para proporcionar accesibilidad a cada dispositivo de la red, el controlador de ONOS permanece lógicamente centralizado y las diferentes instancias separadas de la arquitectura se pueden ver y acceder como en un solo sistema, a través de la interfaz gráfica de usuario (GUI; graphic user interface) que proporciona ONOS.

ONOS sirve distintas APIs en las capas Norte y Sur. Para comunicarse con la capa Norte, ONOS utiliza su subsistema Intent Framework, el cual permite que las aplicaciones especifiquen los recursos necesarios al sistema. Por ejemplo, si una aplicación necesita más ancho de banda. Cuando una aplicación solicita esos recursos el sistema se configura en consecuencia.

Cada instancia de ONOS interactúa con el entorno de red y los dispositivos a través de una API en dirección Sur que se comunica con componentes de nivel inferior. El núcleo descubre qué protocolos se pueden usar para interactuar con el dispositivo, y la API hacia el Sur usa ese protocolo para interactuar con el dispositivo.

ONOS dispone de una gran cantidad de apps con las que gestionar y crear los diferentes flows. No obstante, las que se van a utilizar son la CLOSAPP y la OLTAPP (de las que hablaremos más adelante en este capítulo).

Es importante decir que ONOS funciona en cluster. En versiones anteriores de ONOS, la capacidad de crear y añadir nodos al cluster era responsabilidad del propio ONOS, pero en versiones más recientes (onos 1.14 en adelante) esa responsabilidad se ha escindido del propio ONOS y los desarrolladores han creado otro software de cluster llamado ATOMIX. Este es el encargado de formar y gestionar los nodos que conforman el cluster. Por defecto, cuando arrancamos ONOS este se inicia como un cluster de un solo nodo sin necesidad de tener corriendo ATOMIX en nuestras máquinas.

3.3.2. Niveles e funcionalidad de ONOS y Servicios Primarios

En este punto se describen algunos de los subsistemas que forman parte de ONOS. En la figura 3.10 podemos ver los niveles de funcionalidad divididos en capas. [13].

- Dispositivos: gestiona el inventario de todos los switches de la infraestructura.
- Enlaces: administra el inventario de enlaces de la infraestructura. Un enlace es una conexión directa entre dos puntos.
- Host: dirige el inventario de los nodos de computación (host) y su ubicación en la red.
- Topología: servicio que administra Snapshots de la topología de la red.

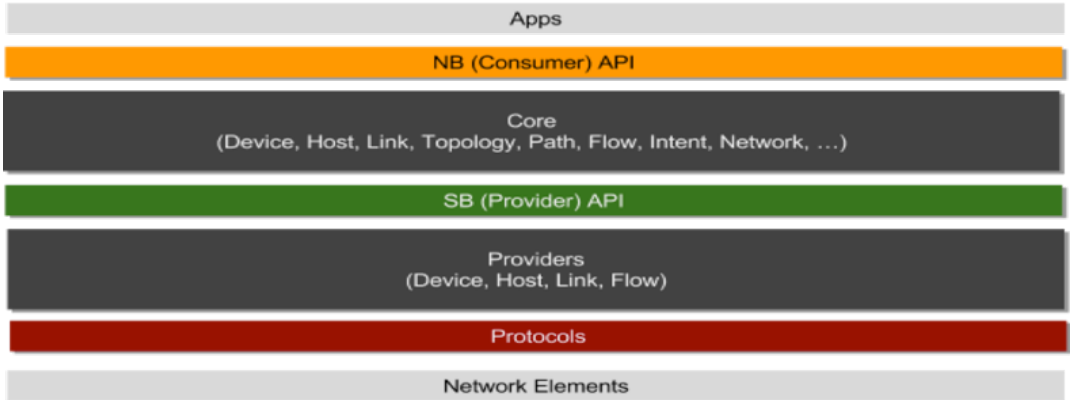

Figura 3.10: Niveles de Funcionalidad de ONOS. Obtenido en [13]

- Servicio de rutas: calcula y encuentra rutas entre los dispositivos de la infraestructura de red usando el Snapshot de la topología más reciente.
- Flows: servicio que gestiona las reglas de flujo instaladas en los dispositivos y proporciona métricas de flujo.
- Paquetes: permite que las aplicaciones escuchen los paquetes de datos recibidos de los dispositivos de la red y emiten paquetes de datos a la red por medio de uno o más dispositivos de la misma.

En la Figura 3.11 se pueden ver algunos de los subsistemas actuales dentro de ONOS. (HOST, TUNNEL, FLOWS, LINKS, etc.)

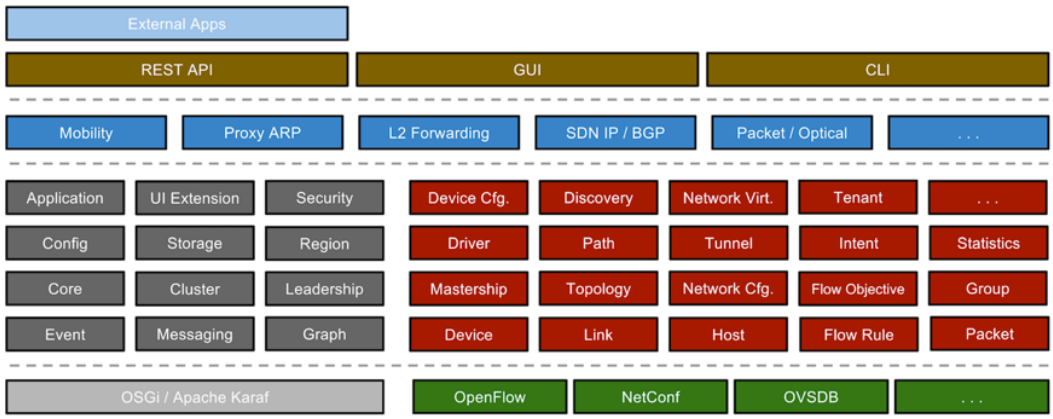

Figura 3.11: Subsistemas actuales de ONOS. Obtenido en [13]

3.3.3. ONOS vs ODL

Muchos son los controladores SDN existentes, pero principalmente los dos más conocidos hoy día son OpenDaylight y ONOS, proyectos de código de abierto y tienen

apoyos de muchas empresas. En el caso de ODL, tiene apoyo de empresas como IBM, Cisco, RedHat, Microsoft, etc. Por su parte, ONOS se encuentra en Samsung, Google, Edge-Core e Intel entre otras.

ODL

OpenDaylight [14] es un controlador SDN de código abierto que tiene 6 años de historia y 8 releases (aproximadamente una release cada 6 meses. Figura 3.12 se aprecia el esquema de ODL.

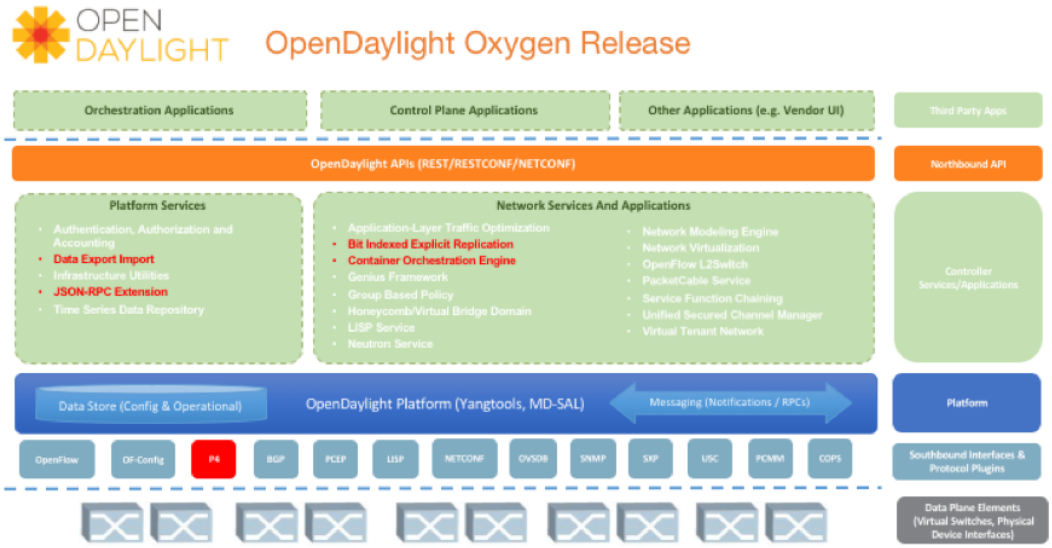

Figura 3.12: Esquema de ODL

El desarrollo y evolución se hace en base a 4 casos de uso principales:

- Nube y NFV
- Delivery Automático de Servicios
- Optimización de Recursos de Red
- Visibilidad y Control

Dentro del desarrollo de ODL, parte de los proyectos son "gestionados" (managed) y se incluyen en las releases oficiales. Otros son "no gestionados" que no se siguen de manera continua. Una posible analogía sería el kernel de Linux, cuya gestión sería similar a la de los proyectos gestionados, mientras que el resto de los elementos de las distribuciones de Linux entrarían dentro de la categoría de "no gestionados".

La mayor parte del uso de OpenDaylight no es del propio controlador, tal y como se distribuye de forma abierta, sino de soluciones basadas en [20] (programa "Powered by OpenDaylight"). Esto conlleva una fragmentación de su uso y desarrollo.

Puntos en común

Tanto ONOS como ODL están escritos en Java, diseñados para uso modular con una infraestructura personalizable y ambos cuentan con los mismos socios casi en su mayoría.

Ambos proyectos son parte de la Linux Foundation y disponen de interfaces Southbound con un número elevado de protocolos, desde OpenFlow, pasando por P4 e incluyendo Netconf.

Diferencias

- Licencia: ONOS tiene una licencia de Apache 2.0, mientras que ODL usa la licencia pública de Eclipse. ONOS está más orientado a las necesidades del proveedor de servicios / proveedor de nube.

- Estructura: ONOS consiste en una serie compleja de subsistemas, así como funciones escalables para sistemas de telecomunicaciones. ODL, por el contrario, utiliza una plataforma de mvc y opera desde una capa de abstracción central.

- Clientes objetivo: ODL y ONOS tienen estrategias comerciales híbridas, pero hay diferencias en cuanto a qué compañía está apelando a qué proveedores. Mientras The Whole Stack conectaba ONOS a las telecomunicaciones, ODL estaba más centrado en los centros de datos.

- Enfoque: ODL se centra en unir el legado (BGP, SNMP, etc.) y NGN (redes de próxima generación OpenFlow y SDN). ONOS se enfcoca más en los aspectos de rendimiento y en la agrupación en clústeres para aumentar la disponibilidad y la escalabilidad, por lo que, resulta de más interés para los operadores. Como resultado, ONOS se centra más en las redes de nivel de operador y las empresas de telecomunicaciones abogan más por sus proyectos. Por otro lado, numerosos proveedores de equipamiento como Cisco, Juniper y NES, están construyendo soluciones basadas en ODL, situación que no se da en el caso de ONOS.

- Abstracción en dirección norte (Intent): ONOS presenta 2 capas de interfaz en dirección Norte: Intent Framework y Vista de red global. Intent Framework protege la complejidad de las operaciones de servicio, permitiendo que las aplicaciones soliciten servicios de red extraídos de los detalles específicos de las operaciones de servicio.

- Casos de uso: ONOS tiene como principal caso de uso CORD (Central Office Re-architected as a Datacentre) desde su inicio, con el objetivo de convertir las centrales en núcleos de procesamiento de datos multiacceso; OpenDaylight ha empezado a soportar recientemente vCO (virtual Central Office), cuyo principal objetivo es virtualizar el acceso fijo.

- Governanza: el principal órgano de gobierno de ODL (TSC) está principalmente compuesto por fabricantes de equipamiento (Ericsson, Lumina Networks) y de software (Red Hat), no habiendo ningún operador en el mismo. En el caso de

ONOS, los miembros del Board [8] son proveedores de servicios y operadores de telecomunicaciones en la mayor parte de los casos.

Los desarrolladores de aplicaciones pueden hacer su trabajo y solo necesitan elevar sus intenciones operativas. ODL se orienta en la misma dirección con el proyecto Network Intent Composition, el cual permitirá a los desarrolladores describir fácilmente sus propios propósitos. ODL espera crear una plataforma de intento uniforme para integrar muchas interfaces de Northbound con intención de usuario. En este aspecto, ODL no se enceuntra tan avanzada como ONOS.

En última instancia, hoy día, ONOS cuenta con la mejor ascendencia y será la mejor opción para las empresas que deseen actualizar radicalmente su modo de acceso a la red (con limitaciones en las experiencias del mundo real).

3.3.4. Aplicaciones ONOS

Como hemos visto ONOS dispone de una serie de aplicaciones con las que hacer uso de todos los servicios que propone. Para nuestra red disponemos de dos aplicaciones propias de Telefónica para hacer uso de los servicios que propone ONOS: CLOSAPP y OLTAPP. Cada una de ellas se encuentra destinada a la instalación de los flows en cada uno de los switches que permitiran la conmutación de paquetes y el tráfico dentro de nuestra red.

CLOSFWD

Esta aplicación será la encargada de comunicar a ONOS cuáles son los flows requeridos para instalar y en qué dispositivos. Esto se consigue a través de una estructura de datos denominada Endpoint. Un Endpoint es una estructura de datos (JSON) en la que definimos unos atributos dependiendo del tipo de Endpoint que necesitemos crear.

Con los atributos definidos en el Endpoint, podremos decirle a onos que instale un flow con las características proporcionadas (mac-origen, mac-destino, device, etc). Una vez tengamos los flows instalados, los switches conmutarán el tráfico de una forma u otra, dependiendo del tráfico y paquetes entrantes.

El funcionamiento de la aplicación es sencillo. A través de la API de la aplicación mandamos el Endpoint que queremos crear. Tenemos muchos tipos de Endpoint, sin embargo para este caso solo se emplearán: volt-endpoint, olt-cliente, olt-control y vpdchost. Algunas de las funcionalidades de las que disponemos son las siguientes:

- Creación de Ednpoints
- Eliminación de Endpoints
- Consulta de los flows instalados
- Consulta del registro de Endpoint

Ejemplo de Endpoint:

```
POST /onos/closfwd-app/closfwdapp/register
Host: <ip nodo onos>:8181
Accept: application /json
Content-Type: application/json
Cache-Control: no-cache

[{
    "@type": "volt",
    "device": "of:0000000000000003",
    "port": "45",
    "vlan": "7",
    "mac":"02:42:ac:44:01:02"
 }]
```

En el capítulo de integración se explicará para qué se usa cada Endpoint. Una vez que se realiza la petición, en primer lugar se procesa comprobando los posibles errores (error en el formato, variables incorrectas, etc); después, la solicitud irá al Manager, y dependiendo del tpo de Endpoint que tengamos, se le pasará al controlador de ese Endpoint. Finalmente ese controlador le dirá al driver de ONOS que instales el Flow.

Cuando realicemos la integración podremos ver un diagrama de secuencia en la creación de los Endpoints.

OltAPP

Esta aplicación es la que usaremos para registrar las OLT que añadamos a nuestra CLOS, de la misma forma que con la aplicaciónde de la CLOS haremos una petición a la API en la que mandaremos la estructura de datos. Este caso no precisa de gran dificultad puesto que la OLT solo tiene dos funcionalidades.

- Registrar la OLT en la clos
- Registrar un Usaurio para la OLT registrada

Para el registro de la OLT le pasaremos el identificados de la OLT y el uplink (puerto lógico por el cual la OLT se conecta a L1).

Endpoint registro olt:

```
POST /onos/ctpd-olt-app/oltapp/register
Host: <ip nodo onos>:8181
Accept: application /json
Content-Type: application/json
Cache-Control: no-cache

[{
 "device": "of:0000000cd5000540",
 "uplink": "129"
}]
```

Para el registro del usuario le pasaremos un puerto, una vlan, y el ancho de banda del que dispondrá.

```
1    POST /onos/ctpd-olt-app/oltapp/register
2    Host: <ip nodo onos>:8181
3    Accept: application/json
4    Content-Type: application/json
5    Cache-Control: no-cache
6
7  [{
8        "bandwidth": "600",
9        "vlan": "0",
10       "port": "66"
11 }]
```

Al contrario que en nuestra app de Clos, en este caso no tenemos Endpoints, si no que nuestras nuevas estructuras son OltDevice y OltVlanPort, haciendo referencia repectivamente a los datos del registro de la OLT y al registro del usuario.

3.4. Esquema de la red y Topología

En la siguiente figura se puede ver el mapa de red que estamos integrando. Por un lado tenemos 6 switches Openflow, dispuestos en una topología CLOS. Cada uno de estos switches se encuentra conectado físicamente a los diferentes componentes que conforman la red. En el switch Leaf 1 (L1) se conecta la OLT y esta a su vez se conecta a una o varias ONT/ONU. Para este supuesto solo usamos un ONT. A la ONT se conectarían los diferentes usuarios de la red. En L2 y L3 estan conectados físicamente dos host (nodos de computación) en los que se almacenan los componentes software de ONOS y VOLTHA. En esos host también estarían los distintos servicios al borde que se quieran ofertar (DNS,DHCP,etc). Aunque para dicho supuesto no se ha realizado ninguna implementación. Por último en L4 tenemos una conexión a una salida a internet. Las diferentes conexiones con los switches no tienen un motivo explícito, simplemente se considera que si los clientes tienen acceso por uno de los extremos, la salida a internet debe encontrarse en el extremo opuesto. En la Figura 3.13 queda representada el esquema completo de la red.

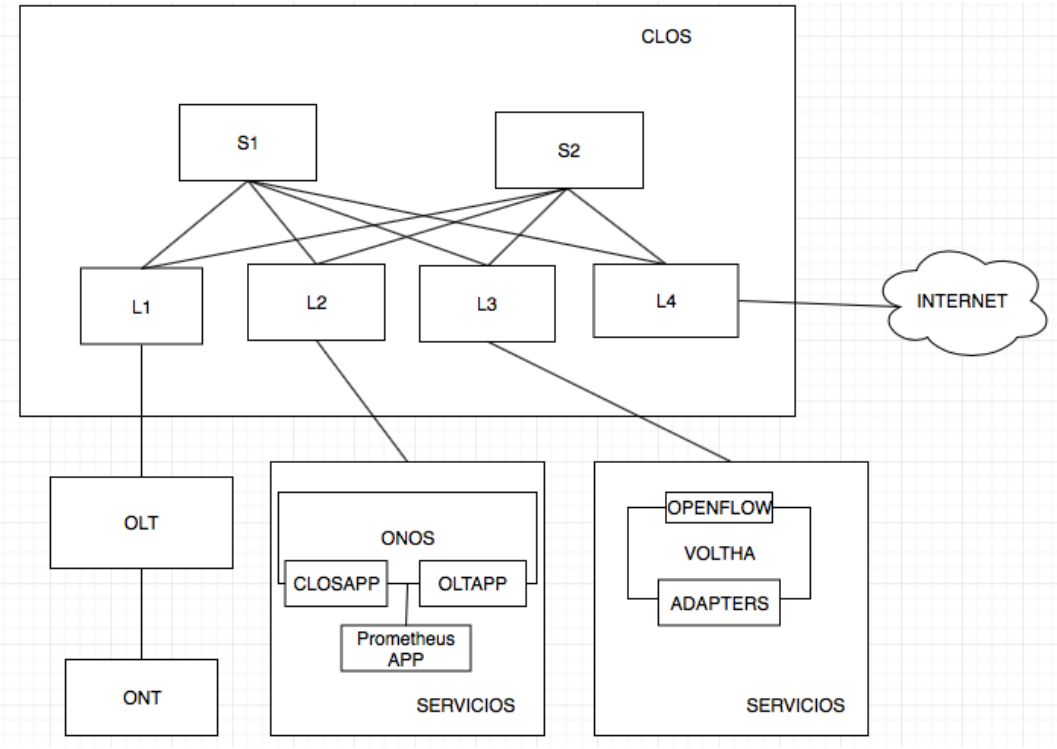

Figura 3.13: Esquema Completo de red

3.4.1. Topología de Red

En esta parte de se explicará la topología de red elegida. En este caso se ha optado por una topología CLOS con 2 switches tipo Spine y 4 switches tipo Leaf con capacidad para escalar horizontalmente En la figura 3.14 se muestran los switches en topología Clos

Topología Clos

Las redes Clos fueron introducidas por Charles Clos [7] a mediados de 1950 como herramienta de cambio en la conmutación de las llamadas. Las redes Clos se convirtieron en topologías de barra transversal y, finalmente, en conmutadores Ethernet basados en chasis utilizando una estructura de conmutación de barra transversal. Actualmente las redes Clos se están utilizando en las arquitecturas modernas de Data Centers para lograr un alto rendimiento y resilencia. Se puede decir que el concepto de redes Clos llevan existiendo desde hace 60 años, pero es ahora cuando más se está utilizando pues resulta clave para las redes de Data Center.

Charles CLos (investigador en los laboratorios Bell) publicó un documento titulado "Estudio de redes de conmutación sin bloqueo" [7]. En este documento se describe cómo las llamadas telefónicas podían conmutarse con equipos que usaban múltiples etapas de interconexión para permitir que se completasen las llamadas. Estos puntos de conmutación se denominan conmutadores de barra cruzada. Las redes Clos fueron diseñadas para ser una arquitectura de 3 capas: una etapa de ingreso, una etapa

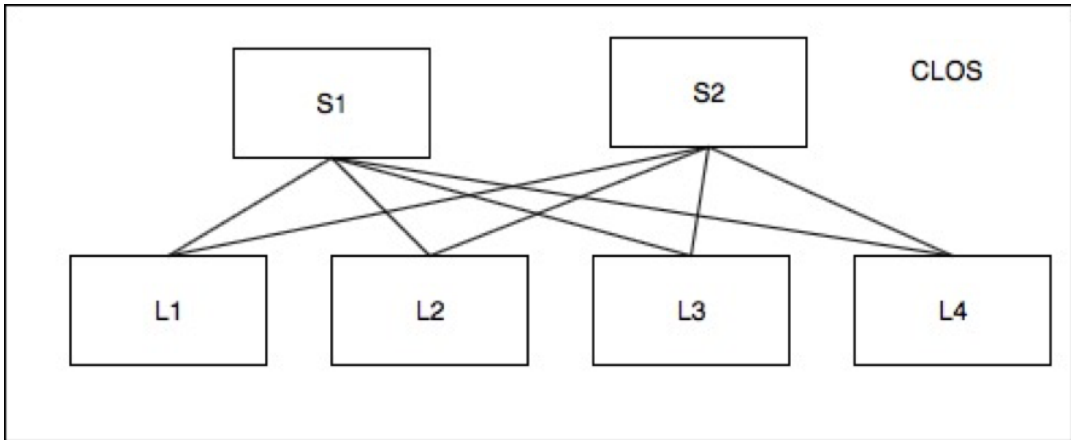

Figura 3.14: Topología CLOS

intermedia y una etapa de egreso. Existen múltiples rutas para que la llamada se cambie a través de la red, de modo que las llamadas están siempre conectadas y nunca bloqueadas por otra llamada.

A lo largo de los años las redes comenzaron a utilizar el modelo de conectividad de "fat tree", basado en una arquitectura de acceso de distribución central. Con el fin de evitar la suscripción excesiva, las velocidades de enlace aumentaron progresivamente a medida que llegaba al núcleo. Por ejemplo, los enlaces de acceso a servidores o equipos de escritorio pueden haber sido históricamente enlaces Fast Ethernet de 100 Mbps; los enlaces ascendentes a los conmutadores de distribución, enlaces Ethernet de 1 Gbps, y los enlaces ascendentes desde allí al núcleo hubieran sido canales de puertos de 4X1Gbps.

El problema de las redes tradicionales creadas utilizando el protocolo de árbol de expansión o las redes centrales enrutadas de capa 3 es que se elige la ruta más óptima del conjunto de caminos posibles, de modo que todo el tráfico elige esta ruta hasta el punto en que se congestiona y se pierden paquetes. El resto de rutas no se usan porque el algoritmo de dicha topología los considera menos deseables o bien porque pueden formas bucles.

Las redes Clos han hecho su reaparición en las topologías modernas de conmutación de centros de datos. Ahora la red clos se manifiesta en la forma en la que los conmutadores están interconectados. El número total de conexiones es el número de conmutadores leaf multiplicado por el número de conmutadores spine (en nuestro caso: 2 X 4 = 8).

Ventajas de esta Topología

En la Clos, cada switch de nivel inferior está conectado a cada uno de los switches de nivel superior en una topología de malla completa. Una de las ventajas de la red Clos es que puede utilizar un conjunto de dispositivos idénticos para crear el árbol y obtener un alto rendimiento y resilencia. Para evitar que se elija una ruta de enlace ascendente, la ruta se elige aleatoriamente para que la carga de tráfico se distribuya uniformemente entre los conmutadores Spine. Si alguno de los conmutadores Spine

falla el tráfico, pasaría por el Spine restante.

Se ha elegido esta topología por ser sencilla, resilente y escalable. Son muchos los DataCenters que están optando por las toplogías Clos, entre ellos Facebook debido a las ventajas que supone y el coste reducido del mismo.

Otras Topologías de Red

A continuación se describen otras opciones de red disponible [10]:

- Hipercubo: una red hipercubo 3D simple es solo un cubo; una caja de seis caras con switches en cada esquina. Figura 3.15, representación de la topología Hipercubo

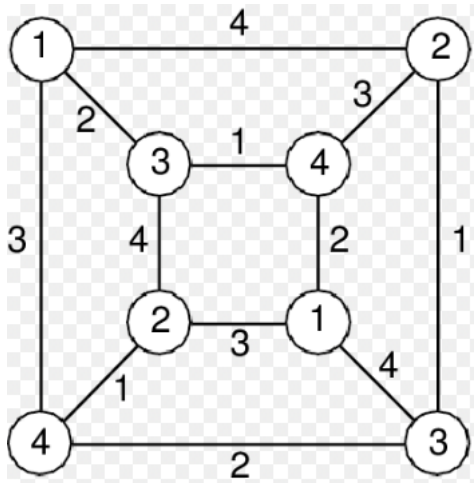

Figura 3.15: Topología Hipercubo

- Toroidal: este término se refiere a cualquier topología en forma de anillo. Un toroide (torus) 3D es una red interconectada de anillos altamente estructurada. Los toroides son una opción popular en los entornos de computación de alto rendimiento. Figura 3.16, representación de la topología Toroidal

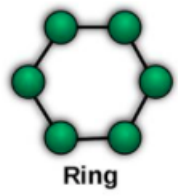

Figura 3.16: Topología Toroidal

- Medusa (Jellyfish): la topología de Medusa es en gran parte aleatoria. En este diseño, los switches están interconectados con base en la preferencia del diseñador de la red. Figura 3.17, representación de la topología Medusa

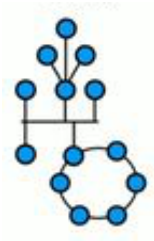

Figura 3.17: Topología Medusa

- DCell: muchos servidores se embarcan con múltiples tarjetas de interfaz de red (NIC). Algunas de estas NIC se conectan en una celda directamente desde un servidor a otro, mientras que otras se interconectan a través de un switch a otras células. DCell asume que un servidor tiene cuatro o más tarjetas de red. Figura 3.18, representación de la topología DCell

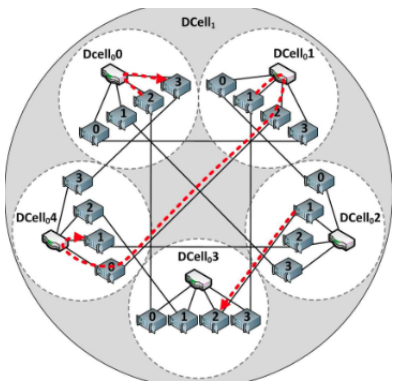

Figura 3.18: Topología DCell

- Mariposa (Butterfly): la Mariposa plana de Google es una construcción de red

específica similar a un tablero de ajedrez. En esta rejilla de switches, el tráfico puede moverse a cualquier switch en una dimensión dada. El objetivo es reducir el consumo de energía. Figura 3.19, representación de la topología Mariposa

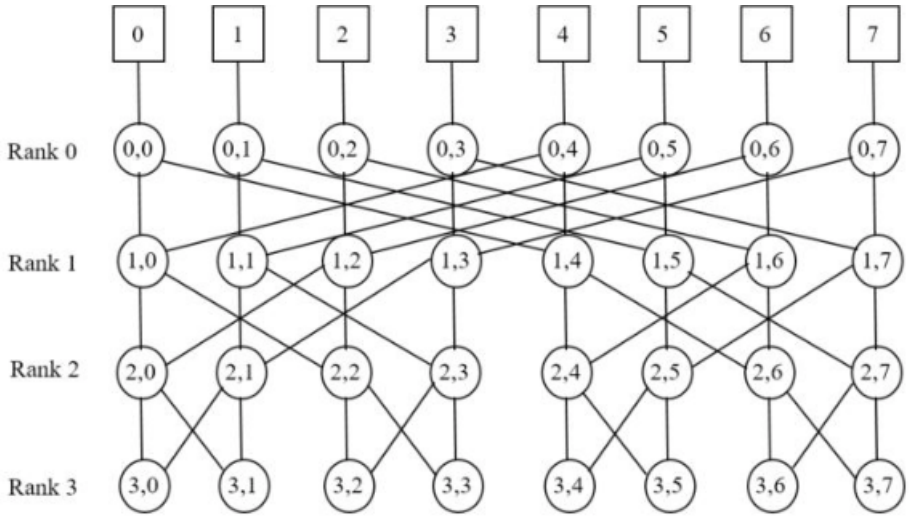

Figura 3.19: Topología Mariposa

Capítulo 4

Integración de los componentes de la red

Este capítulo esta dedicado a la integración y comunicación de cada uno de los componentes que conforman la red. Describiremos el procedimiento de activación y los mecanismos de los que disponemos para verificar su correcta comunicación.

4.1. Activación de los componentes ONOS y VOLTHA

Antes de comenzar con la conexión de las diferentes OLTs, necesitamos tener arrancado el software ONOS y VOLTHA. Actualmente ambos están funcionando como servicios en máquinas virtuales.

4.1.1. Activación de ONOS

Para arrancar ONOS como un servicio necesitamos realizar previamente estos pasos:

En primer lugar, copiamos el script de arranque de ONOS dentro de /etc/init.d

```
1    cp /opt/onos/init/onos.initd /etc/init.d/onos
```

En el caso de que tengamos ONOS en un Ubuntu 16.04, se requerirán los siguientes pasos adicionales: Añadimos el fichero onos.service a /etc/systemd/system para tratar a ONOS como un servicio más.

```
1    cp /opt/onos/init/onos.service /etc/systemd/system/
2    systemctl daemon-reload
3    systemctl enable onos
```

Una vez realizado esto solo necesitamos arrancar ONOS.

```
1    sudo service onos start
```

Tenemos varias formas de interactuar con ONOS: a través de la consola de comandos o el GUI que nos proporciona accesibilidad via web.

Para acceder a la consola de comandos se hace a través de ssh al puerto 8181, con el usuario y contraseña por defecto karaf.

```
ssh karaf@<ip_maquina> -p 8101
```

Al entrar nos encontraríamos con la salida que muestra la Figura 4.1:

Figura 4.1: Consola de comandos ONOS

En un primer momento necesitamos activar la aplicación que implementa el protocolo Openflow, ya que si no es así, no podremos ver ningún switch en ONOS. Los haremos con el siguiente comando:

```
app activate org.onosproject.openflow
```

Una vez arrancada la aplicación deberíamos ser capaces de ver todos los switches que tenemos conectados (L1, L2, L3, L4, S1 y S2) usando el comando "devices". Figura 4.2 salida del comando devices.

Figura 4.2: Switches conectados a ONOS

Los identificadores que se observan en la imagen corresponden a cada uno de los switches. ONOS es capaz de descubrir a todos sus switches a través de una red de gestión. Igualmente, podemos hacer la misma comprobación desde el GUI de ONOS accediendo a través de este enlace:

```
1    http://<IP ONOS>:8181/onos/ui
```

El usuario y contraseña por defecto es onos/rocks. Desde el GUI podemos interactuar con ONOS de una forma más intuitiva viendo los diferentes dispositivos, links y aplicaciones funcionando en ese momento en ONOS. Al ver que tenemos lo switches

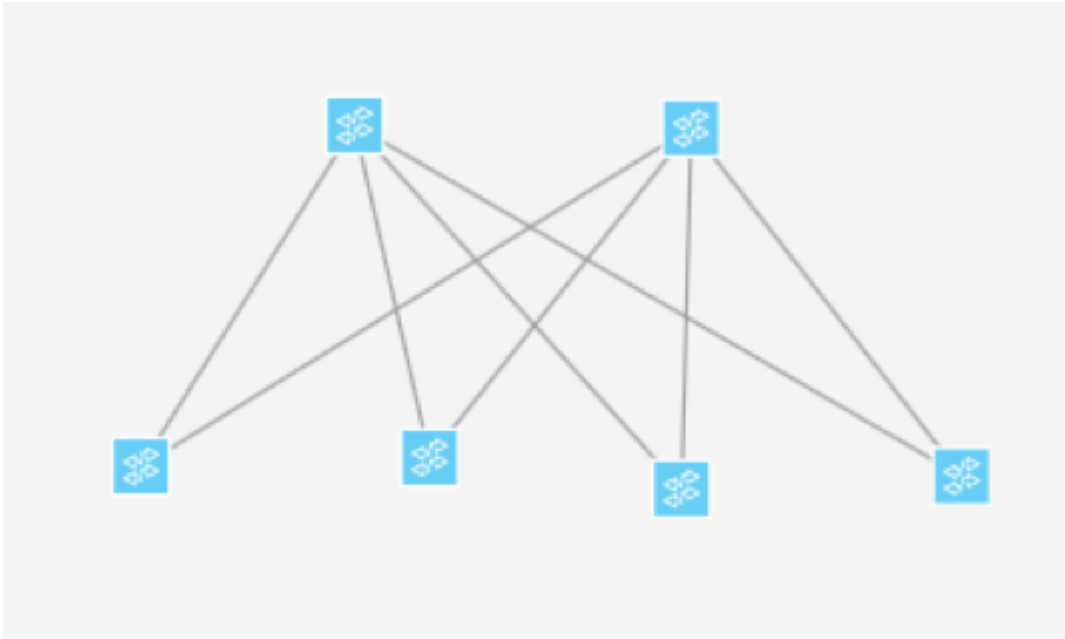

Figura 4.3: Topología de switches visto desde el GUI

conectados y funcionando podemos decir que ONOS se ha iniciado de forma correcta. La figura 4.3 representa la topología clos vista desde el GUI.

4.1.2. Activación de VOLTHA

VOLTHA funciona como un sistema distribuido formado por una serie de contenedores docker que se comunican entre sí, aunque no todos son imprescindibles. Algunos se encargan de generar alarmas o crear gráficas de estadísticas; sin embargo otros, como el propio contenedor de VOLTHA o el contenedor que implenta el agente openflow (ofagent), son estrictamente necesarios. Para este supuesto no hemos eliminado el uso de ninguno de los contenedores.

En primer lugar, antes de arrancar VOLTHA deberemos añadir al archivo de configuración *docker-compose-system-test.yml* la IP de las máquinas donde tengamos arrancados nuestros ONOS. En este compose de docker se muestra la configuración de cada uno de los contenedores. En la configuración del contenedor de ofagent deberemos añadir la IP que tiene asignada ONOS.

```
1    ~/cord/incubator/VOLTHA/compose/docker-compose-system-test.yml
```

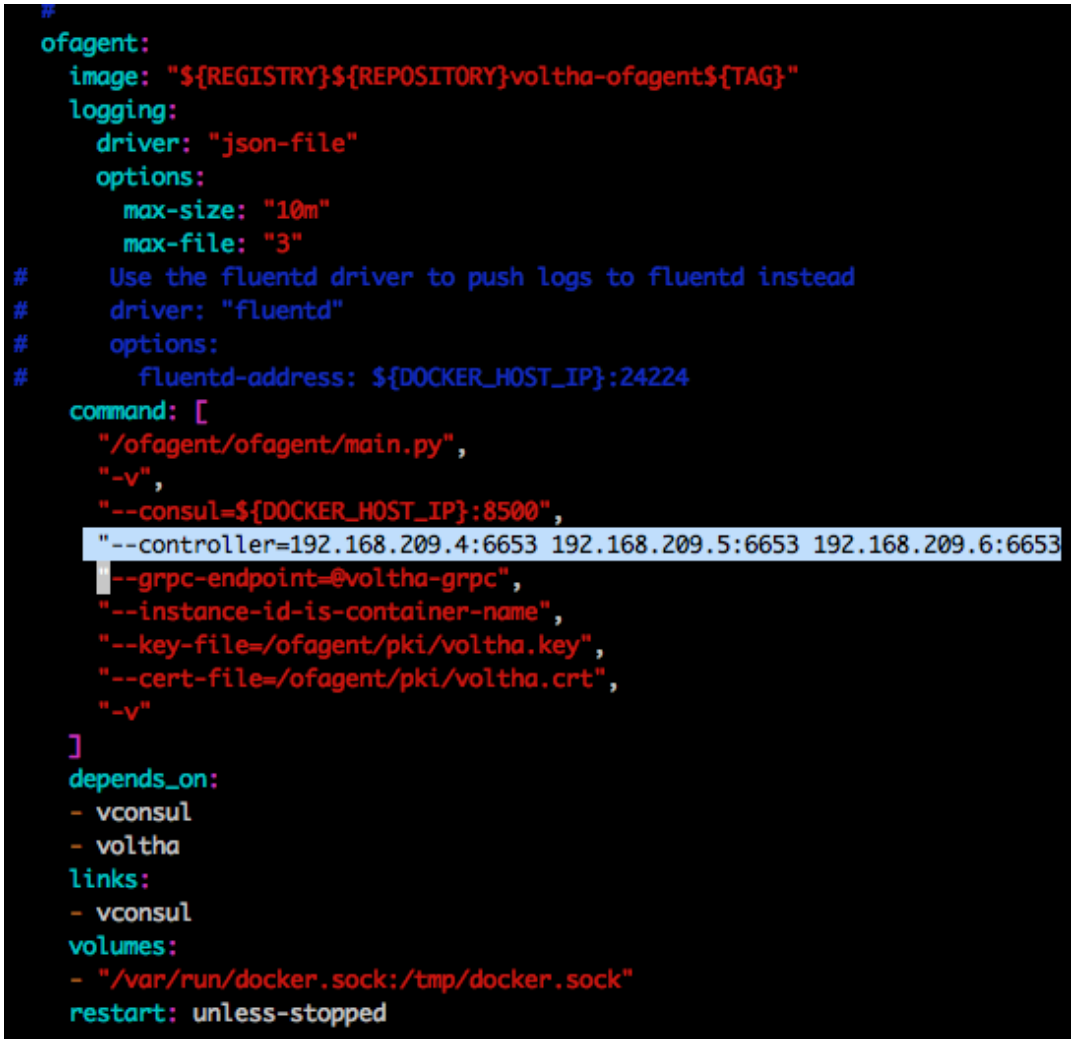

Figura 4.4: docker-compose-system-test.yml

Figura 4.4 muestra la captura archivo de configuración *docker-compose-system-test.yml*.

Una vez añadida la IP a ese archivo debemos asegurarnos de que todos los adaptadores que vayamos a usar estén en el directorio:

```
1    ~/cord/incubator/VOLTHA/VOLTHA/adapters
```

El código de los adaptadores podemos obtenerlo de varias formas: en el repositorio de VOLTHA en GitHub tenemos acceso a los adaptadores disponibles.

Después de estos pasos solo será necesario arrancar VOLTHA con el siguiente comando:

```
1    DOCKERS_UP=$(docker-compose -f compose/docker-compose-system-test.yml ps |
2    grep ' Up ' | wc -l)
```

Si todo se ha iniciado de forma correcta deberíamos ver levantados 15 contenedores docker. La Figura 4.5 es la salida al comando docker-ps que nos muestra todos los

contenedores arrancados. Cada uno de los contenedores desempeña una función.

- Compose-VOLTHA-1: Este es el contenedor principal de VOLTHA, todos los adaptadores específicos del dispositivo son parte de este contenedor
- Compone-nginx-1: NGINX (servidor web/proxy)
- Compose-envoy-1: servicio REST de VOLTHA
- Compose-ofagent-1: Agente Openflow
- Compose-netconf-1: Agente Netconf (Netconf; protocolo de configuración de red)
- Compose-consul-1: Servicio de descubrimiento de contenedores
- Compose-registrator-1: Servicio de directorio de configuración de registro
- Compose-kafka-1: Servicio mensajes bus
- Compose-grafana-1: Servicio de visualización de métricas
- Compose-flientd-1: Servicio para la recolección de datos
- Compose-zookeeper-1: Servicio para la coordinación de procesos distribuidos
- Compose-vcli-1: Servicio de CLI (consola de comandos)
- Compose-dashd-1: Servicio de gestión de dashboards para grafana
- Compose-shovel-1: Servicio de puente para métricas en kafka a graphite
- Compose-portainer-1: Servicio GUI

Ahora que hemos levantado ONOS y VOLTHA es el momento de integrar las OLTs objeto de estudio.

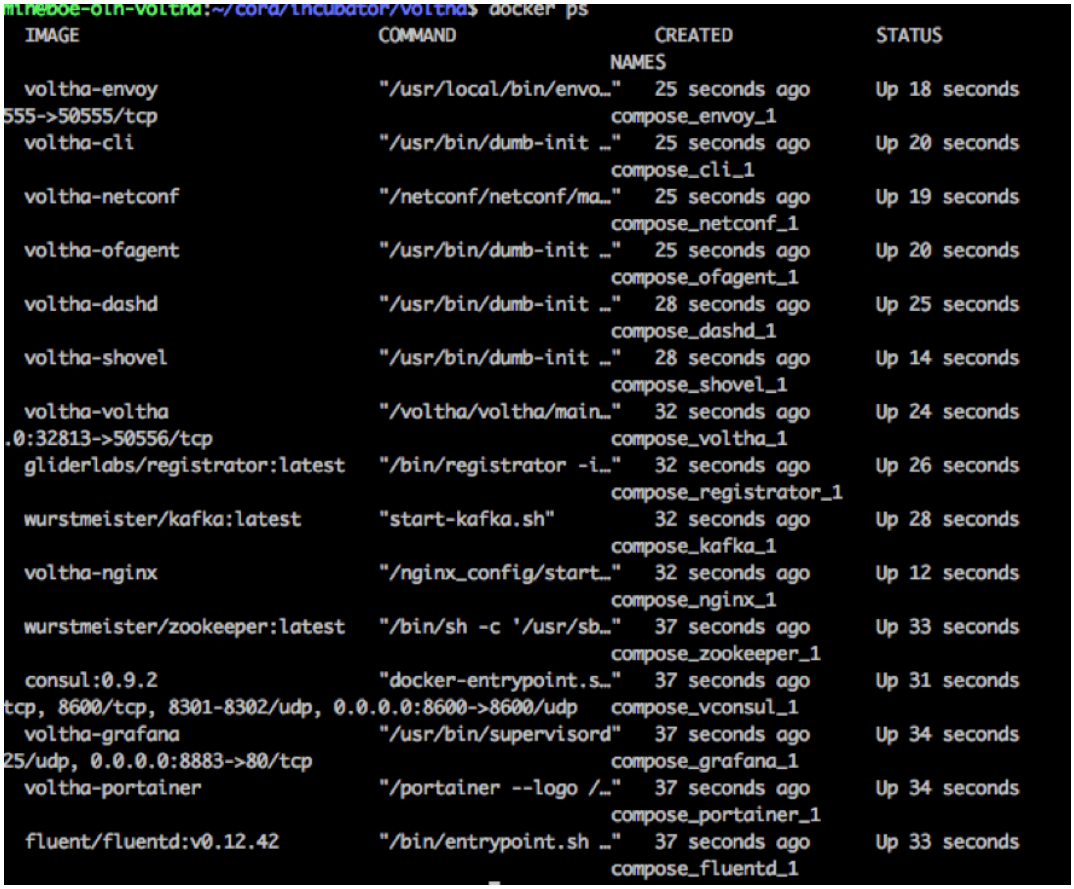

Figura 4.5: Contenedores docker levantados

4.2. Integración OLT GPON Celéstica

Previamente, comenzaremos con la integración de la OLT GPON de Celéstica. Usaremos el adaptador específico microsemi. Comprobaremos que todos los componentes interactuan entre sí y realizaremos una prueba de tráfico para el tramo óptico (Tráfico entre un cliente y un Host)

4.2.1. Pre-provisionar OLT

Primero, necesitamos pre-provionar la OLT. Es decir, necesitamos que la OLT y VOLTHA se comuniquen para que VOLTHA pase a tomar el control de la OLT. La Figura 4.6 se muestra el trayecto ente la comunicación entre la OLT y VOLTHA.

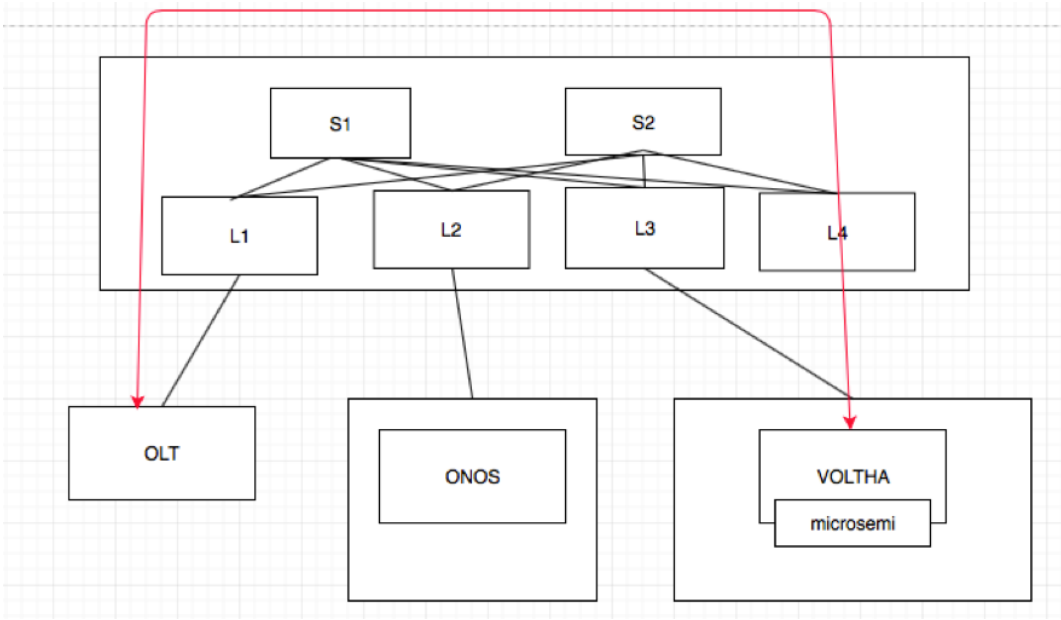

Figura 4.6: Comunicación VOLTHA-OLT

En la siguiente figura se describe el recorrido de las comunicaciones entre ambos actores. Necesitamos que VOLTHA tome control de la OLT y que el tráfico entre la OLT y VOLTHA sea a través de la CLOS. Tanto ONOS como los switches de la CLOS están conectados a una red de gestión. ONOS y los siwtches se ven a través de esta red gestión, pero el tráfico a los servicios en los diferentes Host es el que recorre la CLOS. Esto es importante ya que como hemos explicado, precisamos de un flow que indique el tipo de tráfico entrante (en este caso OLT-VOLTHA) y que permitirá que la red se encargue de gestionarlo. Esto supone que el primer paso implica crear los Endpoints para permitir el tráfico dentro de la red entre la OLT y VOLTHA. Para ello usaremos la aplicación ya mencionada de CLOSAPP.

Creación de Endpoints

Se necesitan crear dos Endpoint para permitir el tráfico dentro de la CLOS: un Endpoint del tipo volt y un Endpoint del tipo olt-control.

Aunque en estas pruebas todos los Endpoints se instauran a mano, en un escenario final se cuenta con un entorno software que crea automáticamente todos los Endpoints necesarios.

```
1   POST /onos/closfwd-app/closfwdapp/register HTTP/1.1
2   Host: <ip nodo onos>:8181
3   Accept: application/json
4   Content-Type: application/json
5   Cache-Control: no-cache
6
7   [{
8       "@type": "volt",
```

```
9            "device": "of:0000000000000003",
10           "port": "45",
11           "vlan": "7",
12           "mac":"02:42:ac:44:01:02"
13       }]
14
```

En el siguiente diagrama podemos ver cuál es el proceso de creación para este Endpoint. Figura 4.7, diagrama de creación Endpoint volt.

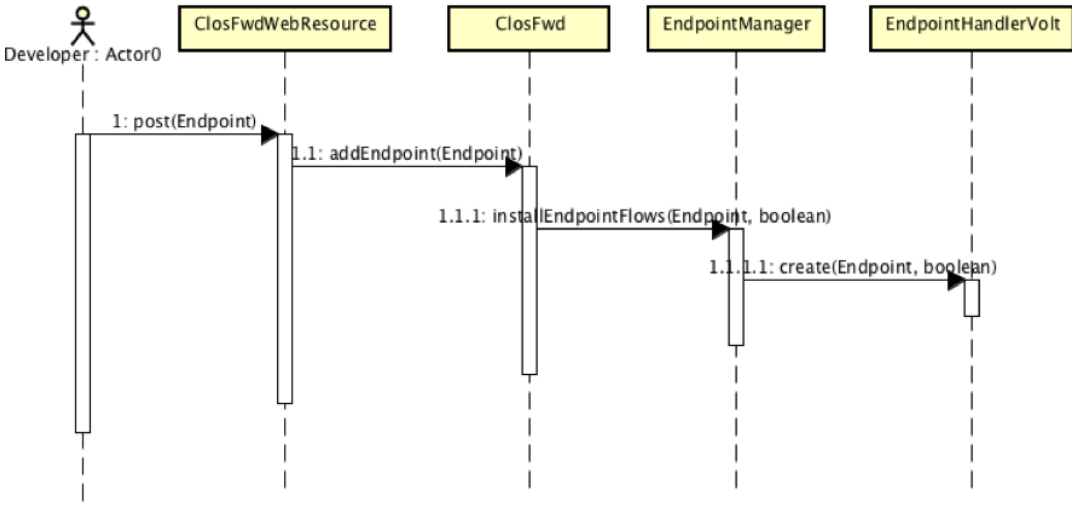

Figura 4.7: Diagrama secuencia creación Endpoint volt

Al crear este Endpoint se nos devuleve la siguiente respuesta.

```
1    {
2      {
3          "@type": "volt",
4          "device": "of:0000000000000003",
5          "mac": "02:42:AC:44:01:02",
6          "port": "45",
7          "ip": "::/0",
8          "vlan": "7",
9          "id": "7b450b09-a0f2-3cf5-b0ff-7e5faf8ee6ee",
10         "reference": 0,
11         "externalAccessFlag": false,
12         "clientAccessFlag": false
13         }
14     }
```

El ID que recibimos es necesario para la creación del segundo Endpoint el olt-control. Ejecutando el comando que mostramos a continuación

```
clos− registritry −list
```

dentro de la consola de ONOS podemos ver un registro de todos los Endpoints que tenemos. Si la petición ha sido correcta nuestro Endpoint deberá aparecer en ese registro. La Figura 4.8 muesrta la salida al comando de consulta de registro de endpoints (clos-registry-list)

Figura 4.8: Resgistro de Endpoint volt

Procedemos con la creación del siguiente Endpoint:

```
http://localhost:8181/onos/closfwd−app/closfwdapp/register
```

```
[{
        "@type": "olt−control",
        "device": "of:0000000000000001",
        "mac": "00:0c:d5:00:05:40",
        "port": "37",
        "vlan": "7",
        " explicit_vlan ":" false",
        " volt_id ":"7b450b09−a0f2−3cf5−b0ff−7e5faf8ee6ee"
 }]
```

En la siguiente figura vemos el diagrama de secuencia para la creación de este Endpoint. El argumento device hace referencia al dispositivo donde se van a instalar los flows, en este caso L1.

En la Figura 4.9 está representada la secuencia de mensajes de creación del EP olt-control

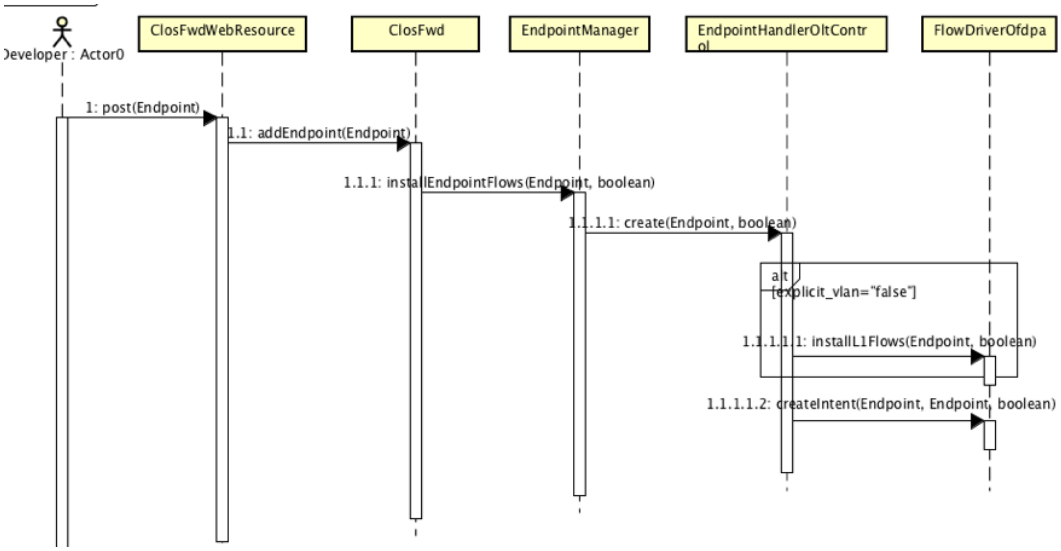

Figura 4.9: Diagrama secuencia creación Endpoint olt-control

Esta es la salida que obtenemos al crear el Endpoint:

```
{
 "89c1e8eb-73c4-303f-baa5-eff21a5151c7": {
      "@type": "olt-control",
      " explicit_vlan ": false ,
      "device": "of:0000000000000001",
      "mac": "00:0C:D5:00:05:40",
      "port": "37",
      "vlan": "7",
      " volt_id ": "7b450b09-a0f2-3cf5-b0ff-7e5faf8ee6ee",
      "id": "89c1e8eb-73c4-303f-baa5-eff21a5151c7",
      "externalAccessFlag": false ,
      " clientAccessFlag ": false
 }
}
```

Comprobamos si tenemos en el registro de Endpoints tenemos uno con el identificador de respuesta. En la figura 4.10 se vemos el Endpoint en el registro de Endpoints

```
UUID:  89c1e8eb-73c4-303f-baa5-eff21a5151c7 Vlan:  7 Mac:  00:0C:D5:00:05:40 Type:  OltControlEndpoint ApplicationId:  512
```

Figura 4.10: Registro Endpoint olt-control

Por último, verificamos que los flows se han añadido de forma correcta con el comando:

```
flows -s -f 000000000000001
```

La cadena numerica hace referencia a un switch de la CLOS con el identificados of:00...X. Podemos ver todos los flows instalados en ese switch. En este caso, L1. En la figura 4.11 se observa la salida al comando flows

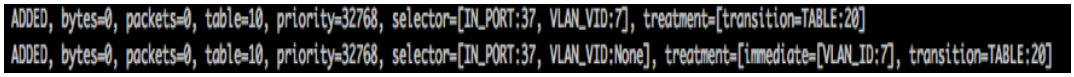

Figura 4.11: Flows olt-control en L1

Con los Endpoints creados y los Flows, ahora es posible conmutar el tráfico entre OLT y VOLTHA, por lo que el siguiente paso es preprovisionar la OLT.

Pre-provisión

Para realizar una preprovisión necesitamos acceder a la consola de VOLTHA de la siguiente forma:

```
1    ~/cord/incubator/VOLTHA$ . env.sh
2    ~/cord/incubator/VOLTHA$ python ./cli/main.py -L
```

al ejecutar ese comando entraremos a la consola de comandos de VOLTHA. En la figura 4.12 observamos que aun no tenemos dispositivos añadidos.

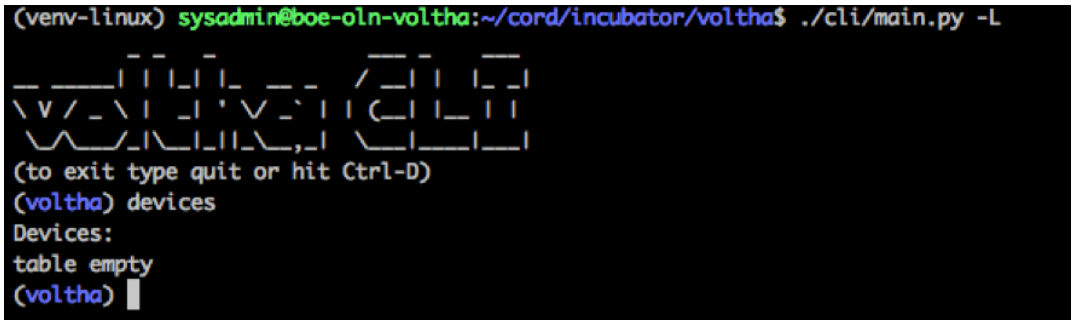

Figura 4.12: Consola de comandos VOLTHA

En la figura podemos ver que acutalmente no hay dispositivos añadidos. El siguiente paso es ejecutar los siguientes comandos para realizar la pre-provisión:

```
1    preprovisin_olt --device-type=microsemi_olt --mac-address=00:0c:d5:00:05:40
2    enable
```

Si la comunicación entre la OLT y VOLTHA ha ocurrido de forma satisfactoria, usando el comando devices deberíamos poder ver nuestra OLT en VOLTHA. La figura 4.13, salida del comando devices con la OLT ya pre-provisionada.

De la misma forma, desde la consola de ONOS también deberemos poder ver

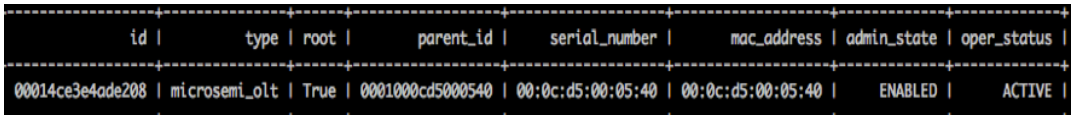

Figura 4.13: OLT añadida a VOLTHA

añadida la OLT como si fuese un switch más. Figura 4.14, salida del comando devices con la ONT ya añadida a VOLTHA

Figura 4.14: OLT añadida a ONOS

4.2.2. Conexión de la ONT

Ahora que hemos conseguido una conexión entre la OLT, VOLTHA y ONOS, el siguiente paso será conectar nuestra ONT a la OLT. Figura 4.15 muestra la comunicación entre la ONT y VOLTHA

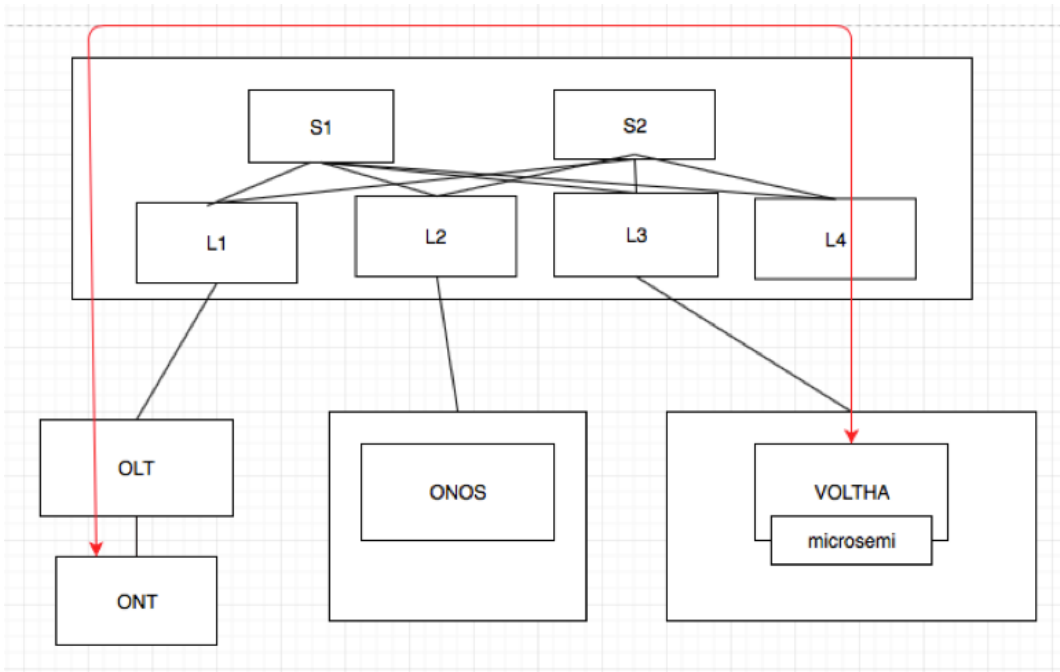

Figura 4.15: Comunicación entre ONT y VOLTHA

Para realizar este paso solo deberemos acoplar el cable de la fibra que tenemos conectada a nuestra OLT a la ONT, y VOLTHA a través de los adaptadores, debería encontrarla y activarla. Para este caso estamos usando una ONT de Mitrastar y el adaptador que usaremos será el brcm-onu.

Si los adaptadores funcionan correctamente, deberemos poder ver nuestro nuevo dispositvo (ONT) en VOLTHA.

Figura 4.16: ONT visible en VOLTHA

4.2.3. Registro de la OLT

Habiendo realizado este paso solo necesitamos registrar la OLT en la CLOS y provisionar a un usuario y de esta forma ya estaremos preparados para comprobar el funcionamiento de la red metiendo tráfico.

En primer lugar, registramos la OLT en la CLOS haciendo uso de la aplicación OLTAPP mencionada anteriormente. Le pasamos la siguiente estructura de datos (OltDevice)

```
1   POST /onos/ctpd-olt-app/oltapp/register
2   Host: <ip nodo onos>:8181
3   Accept: application/json
4   Content-Type: application/json
5   Cache-Control: no-cache
6
7   [{
8       "device": "of:0000000cd5000540",
9       "uplink": "129"
10  }]
```

Device hace referencia a la OLT que queremos registrar y el upkink es el puerto lógico por el cual la OLT está conectada a L1. La figura 4.17 representa la secuencia de mensajes para el registro de la OLT

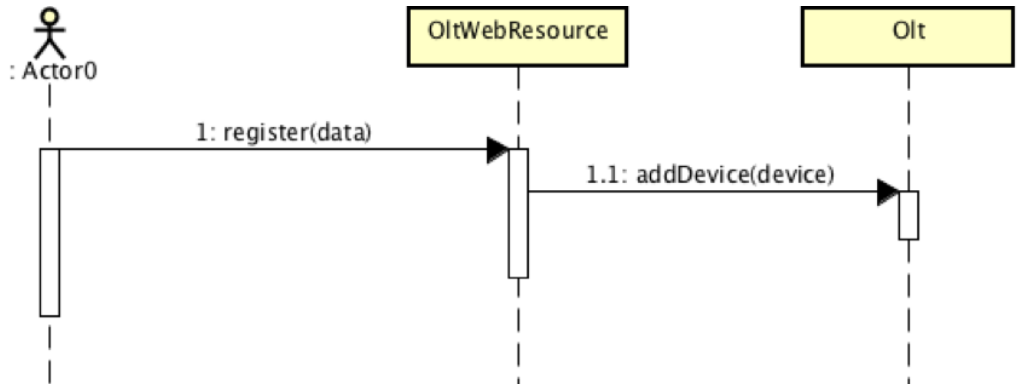

Figura 4.17: Diagrama Secuencia registro OLT

En el tramo óptico hay un doble etiquetado (S-TAG/C-TAG). La que se devuelve es la vlan de la OLT.

Si el registro se produce con éxito, recibiremos una salida identificando la vlan por la que deberá ir el tráfico del cliente.

```
1  {
2      "vlan": 2
3  }
```

Por último, registramos al usuario:

```
1   POST /onos/ctpd-olt-app/oltapp/register
2   Host: <ip nodo onos>:8181
3   Accept: application/json
4   Content-Type: application/json
5   Cache-Control: no-cache
6   [{
7           "bandwidth": "600",
8           "vlan": "0",
9           "port": "65"
10  }]
```

Los datos que pasamos son el ancho de banda que queremos, la vlan de la ONT (la ONT etiqueta con vlan 0 y la OLT desetiqueta esa VLAN y asigna el dobre etiquiteado dependiedno de la ONT) y el puerto por donde va entrar el tráfico (puerto por el que conecta la OLT con la ONT).

4.2.4. Creación de Endpoint Cliente y prueba de Tráfico

El último paso es crear los Endpoints OLT y vpdchost. Este paso es necesario antes de empezar a meter tráfico. El primer Endpoint .ºLT"se instala en L1 para identificar el tráfico que llega de la OLT.

```
POST /onos/closfwd-app/closfwdapp/register HTTP/1.1
Host: <ip nodo onos>:8181
Accept: application/json
Content-Type: application/json
Cache-Control: no-cache

[{
    "@type": "olt",
    "device": "of:0000000000000001",
    "port": "37",
    "vlan": "2"
}]
```

La vlan es por la que irá el tráfico del cliente, la vlan de esa OLT. Al crearlo nos devolverá un id para este Endpoint.

```
{
"776c7d00-0f1a-373d-84e5-3cba5d72c588": {
    "@type": "olt",
    "device": "of:0000000000000001",
    "port": "37",
    "vlan": "2",
    "id": "776c7d00-0f1a-373d-84e5-3cba5d72c588",
    "reference": 0,
    "externalAccessFlag": false,
    "clientAccessFlag": false,
    "mac": "00:00:00:00:00:00"
    }
}
```

El ID que recibimos es necesario proprorcionárselo al siguiente Endpoint que crearemos: el "vpdchost".

```
POST /onos/closfwd-app/closfwdapp/register HTTP/1.1
Host: <ip nodo onos>:8181
Accept: application/json
Content-Type: application/json
Cache-Control: no-cache

[{
    "@type": "vpdchost",
    "device": "of:0000000000000003",
    "port": "45",
    "vlan": "2",
    "ip_client_list": [<lista rango ips cliente>],
    "ip_service_list":[<ips de servicio>],
    "external_access_clients": "true",
```

```
15          " olt_id ": "876d2ec4-9d97-3e88-8fca-d806f976de68"
16      }]
```

En este endpoint añadimos la IP de los clientes y también el ID generado por el anterior Endpoint, ya que el Endpoint de vpdchost es dependiente del Endpoint de OLT. El Endpoint de vpdhost es un conjunto de vpdcs. Cada vpdchost está asociado a una OLT y cada vpdc pertenece a un cliente de esa misma OLT. Un vpdc es un router"para cada cliente.

Haremos una prueba de conectividad para el tramo óptico, mandando el tráfico por un cliente (raspberry) a través de la HGU y comprobaremos si tenemos conectividad con una máquina virtual dentro de un host conectada a L2.

El recorrido que realizará el tráfico es sencillo. El tráfico entra a la red con un doble etiquetado (S-TAG y C-TAG). En este caso, como la vlan de la OTL es la 2 la S-TAG es 2 y la C-TAG es la VLAN del cliente, en este caso la 1 (la vlan se asigna por orden de llegada).

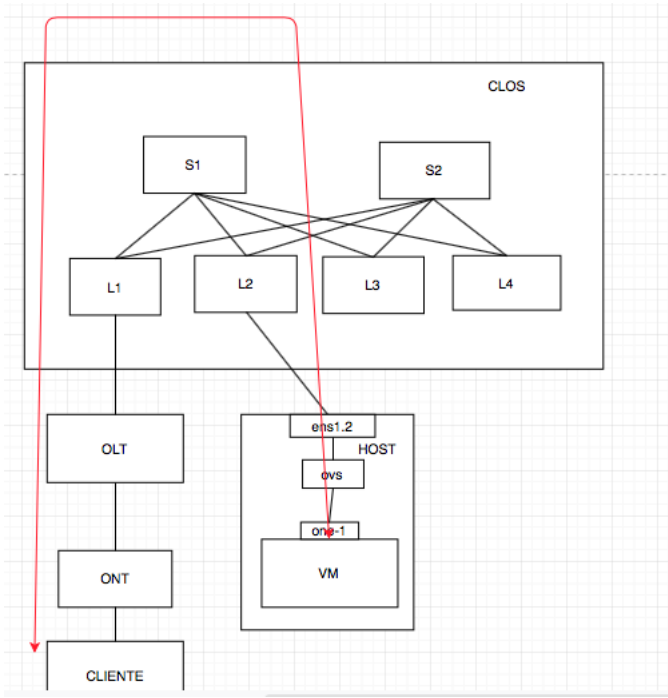

Figura 4.18: Tráfico entre Cliente-VM

La Figura 4.18 muestra la comunicación entre el cliente y el host.

- Entra a L1 con el doble etiquetado (1,2).
- El paquete sube a S1 y posteriormente baja a L2.
- El paquete al tener la vlan exterior 2 entrará por la interfaz de red del host ens1.2 que le quitará la primera VLAN.
- Posteriormente pasa a un OVS (switch virtual) que le quita la segunda etiqueta (la 1).

- El paquete baja hasta la interfaz de la VM.
- La respuesta del paquete sale de la VM sin etiquetar.
- Pasa al OVS donde se le pone la etiqueta 1.
- Sube a la interfaz de red del host ens1.2 donde se le pondrá la segunda etiqueta.
- El paquete sube por L2 a S1 y luego vuelve a bajar a L1.
- El paquete baja por la OLT donde se le quita la vlan 2.
- El paquete baja a la HGU donde se le quita la vlan 1.
- Finalmente recibiremos la respuesta en el cliente.

Para comprobar que esta secuencia se ha completado correctamente, se ha realizado un ping desde una raspberry conectada a la ONT (cliente) hasta una máquina virtual ubicada en el host.

4.3. Integración OLT XGSPON Tibit

Como segundo caso, vamos a integrar la OLT XGSPON de Tibit. Para esta OTL usaremos el adaptador específico de Tibit. Al igual que en el caso anterior, comprobaremos que todos los componentes interactúan entre sí y que podemos transmitir tráfico a través de la red.

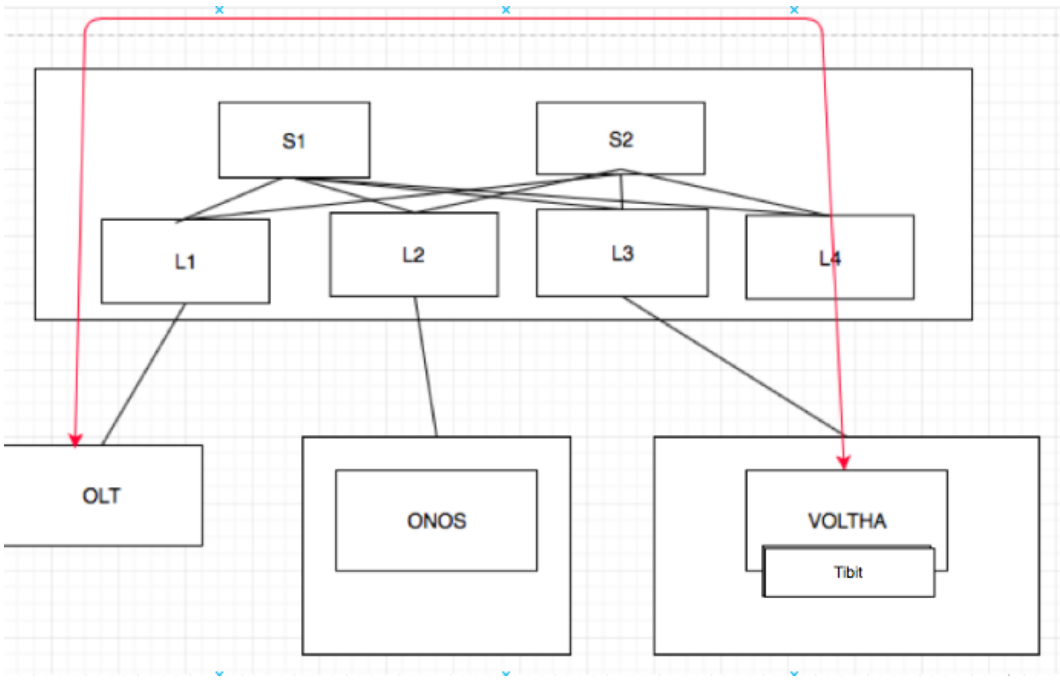

Figura 4.19: Comunicación VOLTHA-OLT

4.3.1. Pre-provisionar OLT

El primer paso es la preprovisión de la OLT. La comunicación entre la OLT y VOLTHA se produce a través de la CLOS al igual que con la OLT de Celéstica. Por lo tanto, antes de realizar la pre-provisión de la OLT es necesario crear los Endpoints.

Creación de Endpoints

Comenzamos con la creación de los Endpoints volt y OLT-control:

```
POST /onos/closfwd-app/closfwdapp/register HTTP/1.1
Host: <ip nodo onos>:8181
Accept: application/json
Content-Type: application/json
Cache-Control: no-cache

[{
"@type": "volt",
"device": "of:0000000000000003",
"port": "40",
"mac": "02:00:5e:b9:85:30",
"vlan": "7"
}]
```

```
POST /onos/closfwd-app/closfwdapp/register HTTP/1.1
Host: <ip nodo onos>:8181
Accept: application/json
```

```
4       Content-Type: application/json
5       Cache-Control: no-cache
6
7       [{
8       "@type": "olt-control",
9       "device": "of:0000000000000001",
10      "port": "47",
11      "vlan": "7",
12      "mac": "70:b3:d5:52:31:31",
13      " explicit_vlan ":"true",
14      " volt_id ":"ced340cf-2936-3a53-8f2b-008a6c3bb8e0"
15      }]
```

Pre-provisión

Seguidamente pre-provisionamos la OLT. Accederemos a la consola de VOLTHA y usaremos los siguientes comandos:

```
1       preprovisin_olt --device-type=tibit_olt --mac-address=70:b3:d5:52:31:31
2       enable
```

La OLT que usamos es de Tibit, por lo que el adaptador también es específico para Tibit. Si la conexión se ha realizado correctamente deberemos de ser capaces de ver la OLT de Tibit en la lista de los dispositivos.

De la misma forma, el dispositivo tambien debe ser visible desde la consola de ONOS.

4.3.2. Conexión de la ONT

El próximo paso es la conexión de la ONT.

Conectamos el cable de fibra que va desde la OLT hasta la ONT. Para esta integración estamos usando una ONT de Alpha y el adaptador de la ONT que usaremos es el brcm-openomci-onu.

Si todo ocurre de forma correcta deberemos ver la ONT en VOLTHA.

4.3.3. Registro de la OLT

Ahora que tenemos conexión entre los diferentes componentes, el siguiente paso será registrar la OLT en la CLOS.

```
1       POST /onos/ctpd-olt-app/oltapp/register
2       Host: <ip nodo onos>:8181
3       Accept: application /json
4       Content-Type: application/json
5       Cache-Control: no-cache
6       [{
7           device    :    of :000170 b3d5523131 ,
8           uplink    :     2
9
10      }]
```

```
{
    "vlan": 4
}
```

Por último, registramos a los usuarios:

```
POST /onos/ctpd-olt-app/oltapp/register
Host: <ip nodo onos>:8181
Accept: application/json
Content-Type: application/json
Cache-Control: no-cache
[{
        bandwidth : 4500 ,
        vlan : 0 ,
        port : 73
}]
```

4.3.4. Creación de Endpoint Cliente y prueba de Tráfico

Finalmente, para la prueba de tráfico en el tramo óptico crearemos los Endpoints de olt y vpdchost:

```
POST /onos/closfwd-app/closfwdapp/register HTTP/1.1
Host: <ip nodo onos>:8181
Accept: application/json
Content-Type: application/json
Cache-Control: no-cache
[{
        @type : olt ,
        device : of:0000000000000001 ,
        port : 47 ,
        vlan : 4 "
}]
```

El identificador que obtenemos es el que debemos añadir a la creación del Endpoint vpdchost:

```
POST /onos/closfwd-app/closfwdapp/register HTTP/1.1
Host: <ip nodo onos>:8181
Accept: application/json
Content-Type: application/json
Cache-Control: no-cache

[{
    "@type": "vpdchost",
    "device": "of:0000000000000003",
    "port": "55",
    "vlan": "4",
    " ip_client_list ": [<lista rango ips cliente>],
    " ip_service_list ":[<ips servicios>],
    " external_access_clients ": "true",
    " olt_id ": "873923akjc4-9r17-4ee58-8zca-d806f912ja1234"
```

```
16      }]
17
18
```

El tratamiento y recorridos que se producen con el tráfico en esta OLT es el mismo que con la OLT de Celéstica.

4.4. Integración OLT XGSPON Edgecore

Como último caso, procederemos con la integración de la OLT de Edgecore haciendo uso de los adaptadores genéricos de OpenOLT y brcm-openomci-onu. Anteriormente hemos explicado cómo funciona openOLT, y en este caso veremos las clases implicadas y secuencia de pasos.

4.4.1. Pre-provisionar OLT

Al igual que en los ejemplos anteriores, lo primero es conectar la OLT con VOLTHA. O lo que es lo mismo: pre-provisionarla. Aunque este caso es un poco distinto, puesto que la comunicación entre la OLT y VOLTHA no circula por la CLOS si no que se realiza de forma directa a través de una red de gestión. La Figura 4.20 muestra el nuevo esquema de comunicación en tre la OLT y VOLTHA.

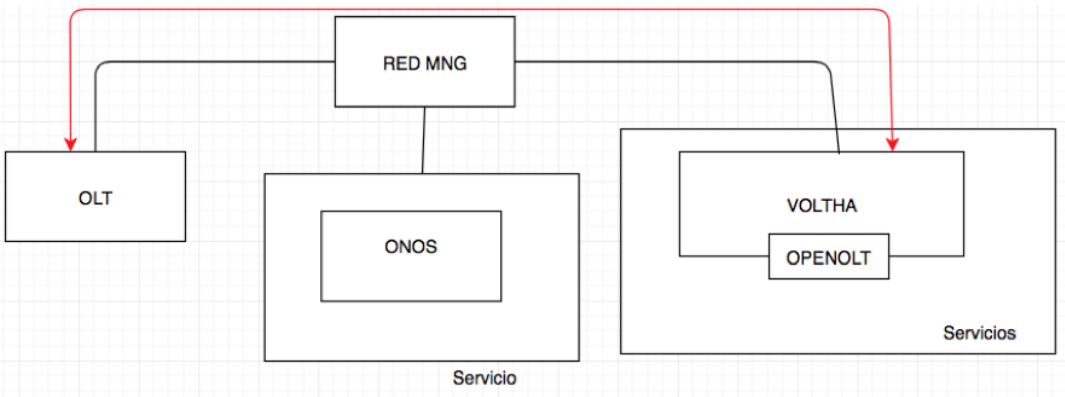

Figura 4.20: Comunicación VOLTHA-OLT Edgecore

Esto quiere decir que no necesitamos crear los Endpoints del tipo volt y olt-control para permitir que circule el tráfico por la CLOS, ya que directamente el tráfico entre OLT y VOLTHA no atraviesa la CLOS.

Iremos a la consola de comando de VOLTHA y usaremos el siguiente comando:

```
1    pre- provisin_olt -t openolt -H 192.168.83.17:9191
2    enable
```

Si el proceso se ha completado con éxito deberemos ver el dispositivo añadido.

De la misma manera, el dispositivo debería poder verse en ONOS:

Figura 4.22: OLT de Edgecore visible en ONOS

Funcionamiento de OpenOLT

En el capítulo anterior hemos explicado qué es OpenOLT y cómo funciona. Ahora comprobaremos cuáles son las clases implicadas en el proceso de pre-provisión. En la siguiente Figura aparecen todas las clases que forman el adaptador OpenOLT y en la Figura 4.24 tenemos el diagrama de secuencia para la pre-provisión. Las clases

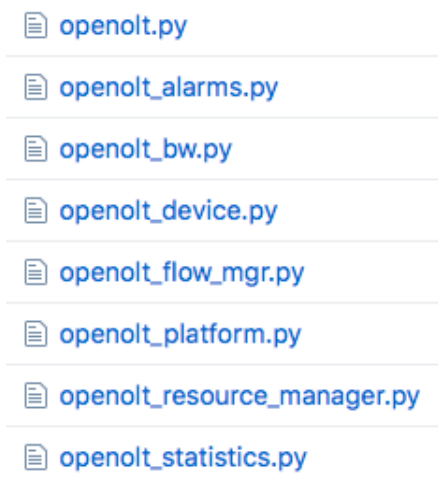

Figura 4.23: Conjunto de clases OpenOLT

involucradas con openolt, openolt_device y openolt_resource_manager.

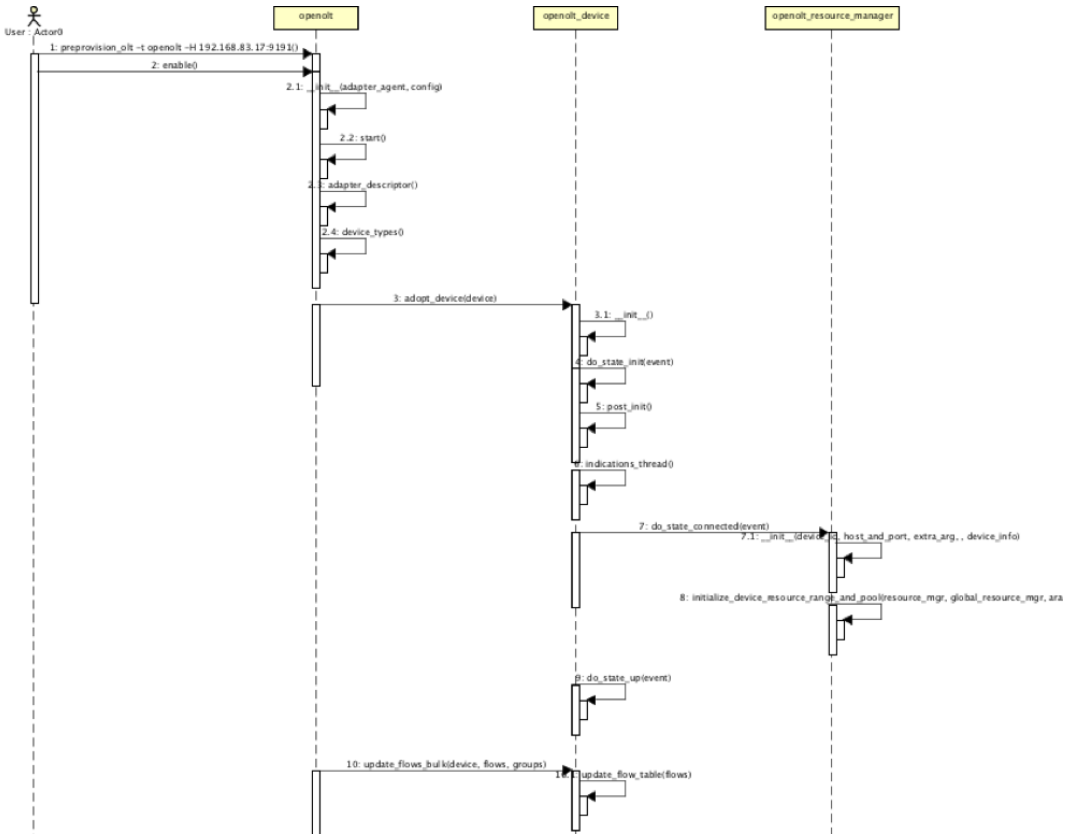

Figura 4.24: Diagrama de secuencia pre-provisión OLT Edgecore

Podemos ver que las clases implicadas son OpenOLT, OpenOLT-device y OpenOLT-resource-manager.

- En primer lugar, se inicia el agente del adaptador. Este paso se inicia después de introducir los comandos en VOLTHA de pre-provisión y enable. Es en este momento cuando comienza la comunicación entre la OLT y VOLTHA.

- Se comprueba el tipo de dispositivo que tenemos. Los datos de la OLT (fabricante, tipo, etc) los proporciona la OLT.

- Se manada un mensaje a openOLT-device para añadir ese dispositivo que ha detectado. Se añade el dispositivo conectado a la lista de dispositivos.

- Se inicia la adopción del dispositivo y se le asigna a un hilo que comprueba el estado periodicamente. Una vez añadido se le asigna a un hilo la tarea de comprobar el estado del dispositivo (conectado, desconectado, alcanzable, etc).

- Se manda un mensaje con un evento a OpenOLT-resource-manager para relizar la conexión con la OLT. Una vez disponemos de los datos de la OLT podemos comenzar con la conexión.

- Se inicializa el dispositivo con unos valores determinados. Los valores como identificador, fabricante y estado se cargar cuando se incializa el dispositivo.

- En OpenOLT-device, después de recibir los valores de la conexión, hace que se ponga el dispositivo como up. Una vez terminada de cargar los valores y la configuración del dispositivo, este pasa a estar activo.

- Por último, la clase OpenOLT manda un mensaje a OpenOLT-device para que instale los flows para la comunicación en ese dispositivo. En caso de que tengamos estos flows previamente añadidos, esta comprobación se hace periódicamente.

- OpenOLT-device actualiza los flows. Al igual que el mensaje anterior este se realiza periódicamente si se encuentra con flows nuevos que instalar o actualizar.

Todos estos pasos en la activación de la OLT van ligados a las respuestas de la OLT. Sin embargo, no es posible comprobar qué clases se ejecutan dentro de la OLT, ya que no tenemos acceso al código del agente OpenOLT de la OLT.

4.4.2. Conexión de la OLT

Después de la conexión entre la OLT y VOLTHA, el siguiente paso es la conexión de la ONT a la OLT.

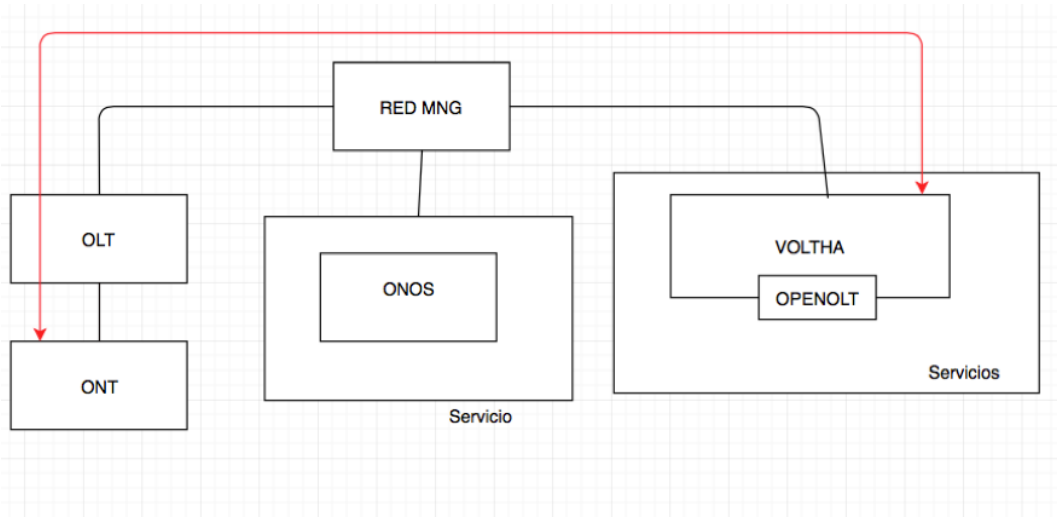

Figura 4.25: Conexión de la ONT con VOLTHA Edgecore

Este paso es sencillo ya que depende solamente del buen funcionamiento de los adaptadores. En este caso, la ONT que usamos es una ONT XGSPON de ALPHA. El adaptador que usaremos es el antes mencionado brcm-openomci-onu.

Ya sabemos cuáles son los estados OMCI. En este punto veremos el diagrama de secuencia para la activación de la ONT usando este adaptador.

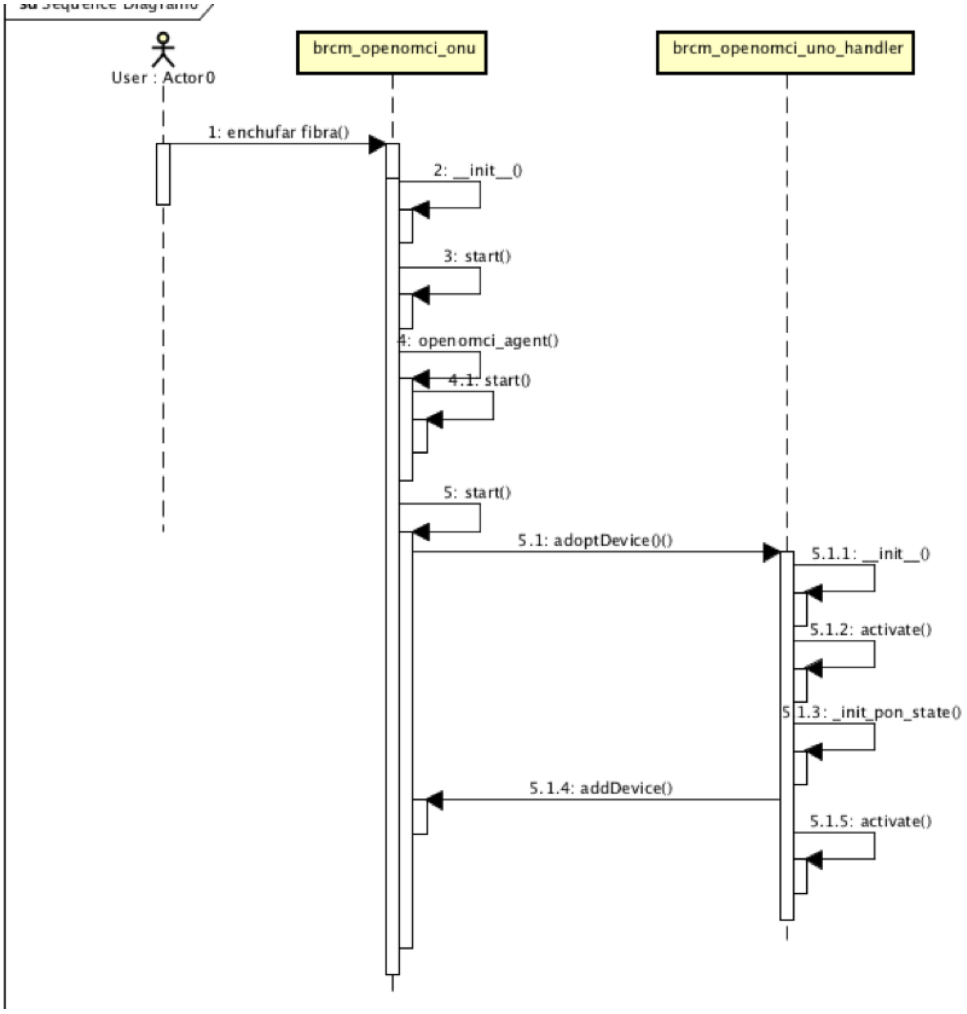

Figura 4.26: Diagrama de secuencia activacion ONT

Observamos las clases implicadas con brcm-openomci-onu y brcm-openomci-onu-handler:

- Primero, al conectar la fibra comiencia con la activación del agente openomci.
- Se le manda un mensaje de adopción de dispositivo a la clase brcm-openomci-onu-handler.
- Brcm-openomci-onu-handler se activa y se inicia el estado pon.
- Le manda un mensaje para añadir el dispositivo a brcm-openomci-onu y después activa dicho dispositivo.

Si todo ha funcionado bien, deberíamos ver los dispositivos en estado ACTIVE en VOLTHA.

Figura 4.27: ONT visible en VOLTHA

Con todo esto, podemos afirmar el éxito de la conexión usando los dos adaptadores genéricos.

4.4.3. Registro de la OLT

Seguidamente deberíamos registrar la OLT en la CLOS, como hemos hecho las dos veces anteriores, poniendo el ID del dispositivo y el uplink.

Llegados a este punto nos encontramos con un obstáculo: el uplink no se ha levantado de forma correcta en la OLT. Esto es debido a un problema con la OLT. Puede ser por parte de la OLT o el propio agente de la OLT. Se precisa de soporte para solucionarlo. La integración queda anclada en este punto debido a un bloque de terceros que impide el avance.

4.4.4. Pasos siguientes

Los pasos siguiente serían los anteriores realizados para las otras OLT:

- Registrar la OLT
- Registrar al usuario
- Crear los Endpoint para permitir el tráfico dentro de la red (olt/vpdchost)
- Generar tráfico y comprobar la conectividad

También se habría analizado cómo gestiona el adaptador los flows según los vamos instalando en la OLT.

Capítulo 5

Monitorización de la red

En el siguiente capítulo se abordará la monitorización de la red, las herramientas desarrolladas y utilizadas.

Una de las complejidades que presentan las SDN es la monitorización de los diferentes componentes que conforman la red. Usaremos un patrón de monitorización compuesta.

La monitorización compuesta es el primer patrón de monitorización moderno. Este patrón consiste en el uso de herramientas especializadas, formando así una "plataforma"de monitorización.

5.1. Estado Actual de la monitorización

La monitorización ha sido un tema desatendido a lo largo del tiempo[16Refenrecia]. En 2011 se produjeron conversaciones en Twitter sobre por qué la monitorización era tan deficiente en torno al hashtag *#monitoringsucks*. Esto se convirtió en *#monitoringlove* y en la fundación de la conferencia Monitorama en Boston, donde se habló sobre todo de cómo mejorar las cosas. Uno de los mayores temas planteados fue la necesidad de crear herramientas nuevas y mejores, mucho más especializadas. De este modo nació la monitorización compuesta como una idea, y se ha convertido en un estándar *de facto* en la práctica.

Una de las ventajas más importantes de este mecanismo es que no es necesario el uso de una única herramienta a largo plazo. Si una herramienta ya no satisface unas necesidades, se puede eliminar y reemplazar por otra, en lugar de reemplazar toda la plataforma.

5.1.1. Componentes de un Servicios de Monitorización

Si vamos a construir una plataforma de monitorización a partir de componentes especializados , primero tenemos que nombrar los puntos de un sistema de monitorización. este servicio se divide en 5 puntos:

1. Recopilación de datos

2. Almacenamiento de datos

3. Visualización

4. Análisis y reportes

5. Alertado

Para cada uno de estos puntos se está haciendo uso de una herramienta que funciona junto con otras para formar un sistema de monitorización completo.

5.2. Recopilación de datos

Tenemos varios componentes que debemos monitorizar (ONOS, VOLTHA, switches, servicios, etc.) Para este supuesto nos centraremos en la monitorización de ONOS y VOLTHA.

Para la recopilación de datos he desarrollado una App (Python) que realiza consultas a las REST API de ONOS y VOLTHA pidiendo diferentes datos.

Estas API nos proporcionan una gran cantidad de datos que podemos obtener. Hemos desarrollado diferentes scanners para cada consulta al API. Cada scanner realiza una consulta específica a los elementos que forman parte de ONOS. Cada scanner no solo tiene la funcionalidad de recopilar datos y para despues almacenarlos, esos datos son tratados (cambiando formado, mapeando valores, etc) lo que lo hace una herramienta de recopilación muy flexible ya que permite incluso la generación de datos nuevos con los datos que estamos obteniendo y después almacenarlos.

Hacemos una consulta de los siguientes elementos: flows, devices, links y ports. Cada consulta nos proporciona una colección de datos JSON, de los cuales elegimos los datos que consideramos más relevantes para la monitorización.

- Flows: conjunto de todos los flows instalados en los swtiches y sus características.

```
1   GET /onos/v1/flows
2   Host: <ip nodo onos>:8181
3   Accept: application/json
4   Content-Type: application/json
5   Cache-Control: no-cache
6 {
7   "flows": [
8     {
9       "id": "281476516477510",
10      "tableId": "60",
11      "appId": "org.onosproject.core",
12      "groupId": 0,
13      "priority": 40000,
14      "timeout": 0,
15      "isPermanent": true,
16      "deviceId": "of:0000000000000004",
17      "state": "ADDED",
18      "life": 81929,
19      "packets": 5901,
20      "bytes": 2012241,
```

```
            "liveType": "UNKNOWN",
            "lastSeen": 1559029452044,
            "treatment": {
                "instructions": [
                    {
                        "type": "OUTPUT",
                        "port": "CONTROLLER"
                    }
                ],
                "deferred": []
            },
            "selector": {
                "criteria": [
                    {
                        "type": "ETH_TYPE",
                        "ethType": "0x88cc"
                    }
                ]
            }
        },
```

De todos los datos que obtenemos, seleccionamos el ID, los packets y los bytes para almacenarlos. Con estos datos pretendemos analizar el comportamiento del sistema analizando:

- Calcular la cantidad media de tráfico que circula por ese flow.
- Saber si algún componente se ha visto afectado si dejamos de ver tráfico por el flow mencionado.

- Devices: conjunto de los dispositivos (switches) y sus características.

```
GET /onos/v1/devices
Host: <ip nodo onos>:8181
Accept: application/json
Content-Type: application/json
Cache-Control: no-cache
{
"devices": [
    {
        "id": "of:0000000000000004",
        "type": "SWITCH",
        "available": true,
        "role": "STANDBY",
        "mfr": "Broadcom Corp.",
        "hw": "OF-DPA 2.0",
        "sw": "OF-DPA 2.0",
        "serial": "",
```

```
17              "driver": "ofdpa3",
18              "chassisId": "4",
19              "lastUpdate": "1558947526421",
20              "humanReadableLastUpdate": "connected 22h51m ago",
21              "annotations": {
22                  "channelId": "192.168.83.14:57974",
23                  "managementAddress": "192.168.83.14",
24                  "protocol": "OF_13"
25              }
26          },
```

De esta consulta seleccionamos para almacenar *available* y *lastUpdate* (lastUpdate se obtiene en formato epoch. Hay que transformarlo a un formato apto para la lectura) Almacenando estos datos queremos analizar lo siguiente:

- El estado de los dispositivos. Si algún dispositivo se desconecta podremos generar alarmas para avisarnos.
- Cada cuánto cambia el estado en los dispositivos.

- Ports: estadísticas sobre los puertos.

```
1     GET /onos/v1/statistics/ports
2     Host: <ip nodo onos>:8181
3     Accept: application/json
4     Content-Type: application/json
5     Cache-Control: no-cache
6  "statistics": [
7          {
8              "device": "of:0000000000000004",
9              "ports": [
10                 {
11                     "port": 1,
12                     "packetsReceived": 1063473,
13                     "packetsSent": 1522442,
14                     "bytesReceived": 124629690,
15                     "bytesSent": 842444432,
16                     "packetsRxDropped": 169477,
17                     "packetsTxDropped": 0,
18                     "packetsRxErrors": 0,
19                     "packetsTxErrors": -1,
20                     "durationSec": 0
21                 },
```

De esta consulta seleccionamos todos los datos menos el campo durationSec, con los que analizaremos:

- El estado de los puertos. Ante cambios bruscos en el tráfico de datos podremos saber si algún componente ha fallado.

- Cantidad de dato que pasan por los puertos y elaboración de estadísticas con ellos.

- Links: conjunto de conexiones entre los diferentes dispositivos.

```
GET /onos/v1/links
Host: <ip nodo onos>:8181
Accept: application /json
Content-Type: application/json
Cache-Control: no-cache
 "links": [
    {
        "src": {
            "port": "5",
            "device": "of:0000000000000f02"
        },
        "dst": {
            "port": "5",
            "device": "of:0000000000000002"
        },
        "type": "DIRECT",
        "state": "ACTIVE"
    },
```

Con este scanner realizamos dos consultas: una a los links y otra a las estadísticas de los puertos para conocer, no solo el estado de link, si no también el estado de los puertos que forman parte de ese link. Con los datos recogidos pretendemos analizar:

- El estado de los links. Si algún enlace se cae, generaremos alarmas que nos avisará.

- El estado de un puerto de un link. En el caso de que un link no esté caído, pero tampoco tengamos tráfico, podremos ver el estado de los puertos que forman parte de ese link y analizar qué ocurre.

No todos los puertos forman parte de un link, por eso es necesario una consulta previa para saber las estadísticas de todos los puertos. Con la consulta a los links podemos saber el estado concreto de los puertos que forman parte de esos links.

5.3. Almacenamiento de datos

El almacenamiento de datos se realiza usando una de las herramientas de monitorización que nos proporciona el entorno Prometheus.

5.3.1. Prometheus

Es un conjunto de herramientas de monitorización [9] y alertado de sistemas Open Source construido por SoundCloud. Es un proyecto que se mantiene independientemente de cualquier compañía.

Las características principales de Prometheus son las siguientes:

- Un modelo de datos multidimensional con datos de series de tiempo identificados por nombre de métrica y pares clave/valor.
- PromQl, como lenguaje de consulta.
- Nodos de servidor autónomos.
- Múltiples modos de gráficos.

Algunos los componentes que tiene Prometheus son su servidor principal el cual almacena los datos, bibliotecas para desarrollar código, exporters para diferentes servicios como HAProxy o Graphite, un administrador de alertas y varias herramientas de apoyo.

En la siguiente figura se muestra un esquema de la arquitectura de Prometheus.

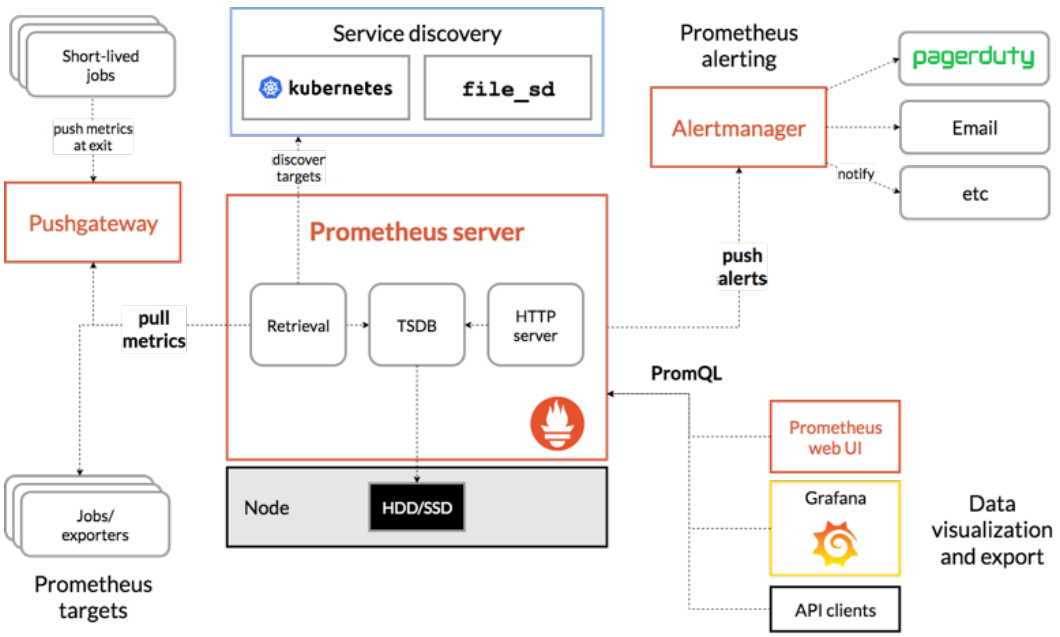

Figura 5.1: Arquitectura Prometheus, obtenido de [9]

5.4. Visualización y Análisis

Prometheus viene incorporado con un servicio para la visualización de los datos almacenados, aunque se ha optado por una opción más especializada para la visualización de métricas: Grafana. También usamos Grafana para el análisis de las métricas.

5.4.1. Grafana

Grafana [6] es un sistema de análisis y visualización de métricas que funciona con múltiples orígenes de datos, entre ellos Prometheus, aunque también nos encontraríamos con Loki, MySQL, Graphite, etc.

Grafana nos proporciona una serie de servicios, entre ellos el de Alertado, al igual que Prometheus.

5.5. Alertado

Para generar alertas se está haciendo uso de dos herramientas: una de ellas es el servicio de alertado que proporciona Grafana, y el otro es una de las herramientas que incluye Prometheus, Alert-manager. Para este supuesto solo haremos uso del sistema de alertado que nos proporciona Grafana

5.6. Funcionamiento

En la Figura 5.2 se aprecia un esquema de cómo están conectados los componentes del sistema de monitorización. Por un lado tenemos los exporters (aplicaciones que

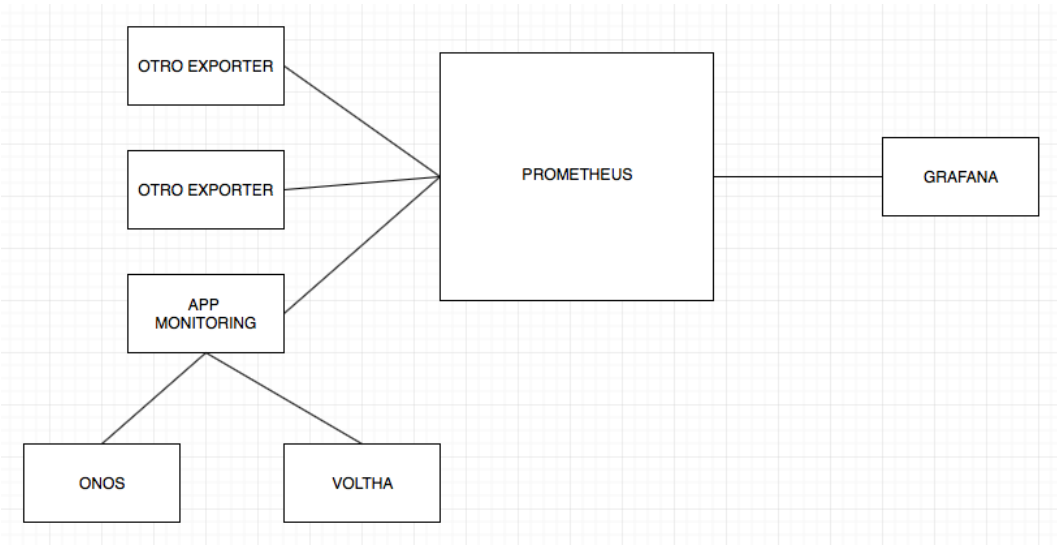

Figura 5.2: Esquema conexión de componentes

recogen y transmiten datos), estando entre ellos la aplicación que se ha desarrollado. Esos exporters se conectan al servidor de Prometheus, donde se almacenan los datos. Grafana se conecta al servidor de Prometheus para recoger esos datos y visualizarlos.

Podemos acceder al servidor de Prometheus vía web y comprobar desde dónde está recibiendo datos. En la Figura 5.3 observamos los targets (exportadores de datos) que están mandando datos a Prometheus y en la Figura 5.4 observamos el target de nuestra app.

```
1   http://<ip maquina prometheus>:8181/targets
```

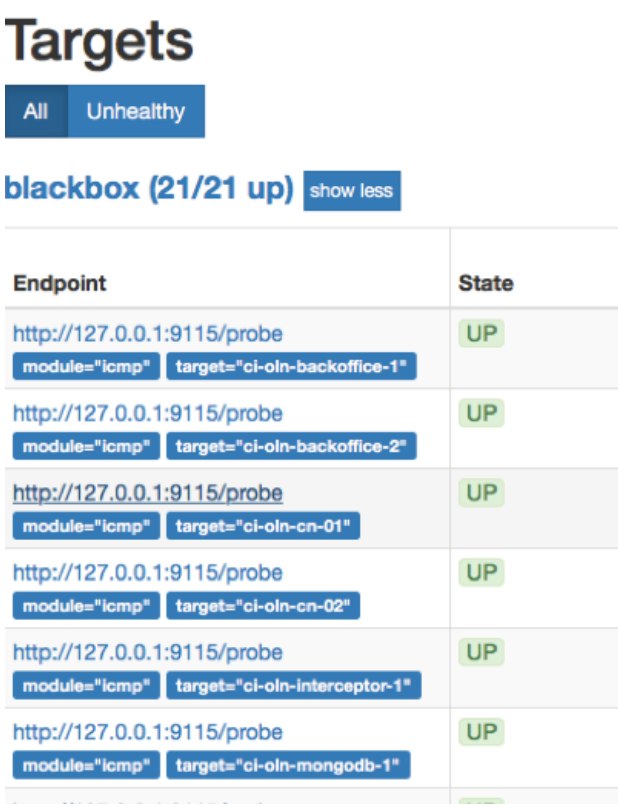

Figura 5.3: Target de la app

Uno de los targets activos será nuestra App.

Figura 5.4: Targets de prometheus

Accediendo via web a

```
1   http://<ip app monitoring>:<puerto publicado>/metrics
```

```
2
```

Se pueden comprobar los datos que estamos exportando. La figura 5.5 muestra los datos que estamos recogiendo, sin aplicar ningún tipo de filtrado.

```
# HELP onos_bytesSent onos_bytesSent
# TYPE onos_bytesSent gauge
onos_bytesSent{device="of:000000000000f01",port="70"} 0.0
onos_bytesSent{device="of:0000000000000004",port="69"} 0.0
onos_bytesSent{device="of:0000000000000004",port="97"} 0.0
onos_bytesSent{device="of:0000000000000001",port="59"} 0.0
onos_bytesSent{device="of:000000000000f02",port="95"} 0.0
onos_bytesSent{device="of:0000000000000001",port="10"} 0.0
onos_bytesSent{device="of:0000000000000003",port="57"} 0.0
onos_bytesSent{device="of:000000000000f02",port="5"} 2.3597054e+08
onos_bytesSent{device="of:0000000000000002",port="44"} 0.0
onos_bytesSent{device="of:000000000000f01",port="30"} 0.0
onos_bytesSent{device="of:0000000000000003",port="5"} 0.0
onos_bytesSent{device="of:0000000000000002",port="5"} 2.30948743e+08
onos_bytesSent{device="of:0000000000000002",port="100"} 0.0
onos_bytesSent{device="of:0000000000000003",port="102"} 0.0
```

Figura 5.5: Targets de prometheus

Estos son algunos de los datos que estamos recibiendo. En concreto, los bytes mandados por cada uno de los puertos.

A partir de estos datos elaboraremos las métricas de visualización en Grafana.

5.6.1. Dashboards Grafana

Los targets de Prometheus nos muestran los orígenes desde donde estamos recogiendo los datos. Grafana nos permite seleccionar a Prometheus como fuente de datos para crear sistemas de visualización (grácficas, tablas, etc).

Para este supuesto vamos crear una tabla donde veremos los links de ONOS y su estado.

En primer lugar necesitamos acceder a Grafana vía web:

```
1   <ip maquina grafana>:8080/grafana
```

La primera vez que entramos en Grafana, vemos la salida que muestra la siguiente Figura. En el panel elegiremos la opción *Choose Visualization* y se nos mostrarán los

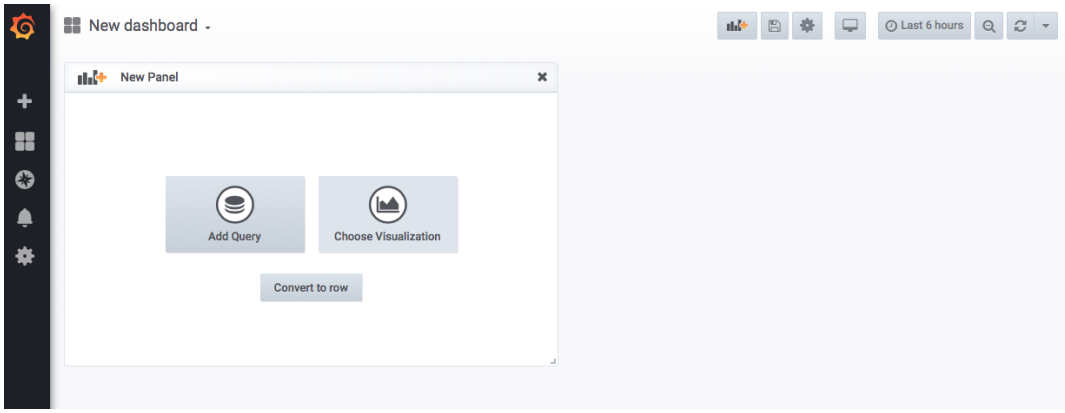

Figura 5.6: GUI Grafana

sistemas de visualización que tiene ofrece Grafana.

Para este supuesto elegiremos la tabla como medio de visualización.

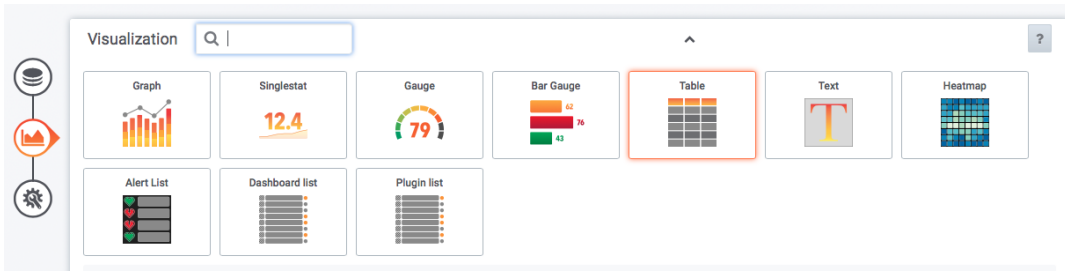

Figura 5.7: Opciones de sistemas de visualización Grafana

El siguiente paso será elegir el origen de datos y los datos que queremos representar.

En la siguiente Figura se muestra la elección de los datos que vamos a representar y el origen de datos.

Con la siguiente sentencia elegimos la métrica onos_link y dentro de esos datos seleccionamos que tengan el estado ACTIVE

```
1   onos_link(state="ACTIVE")
```

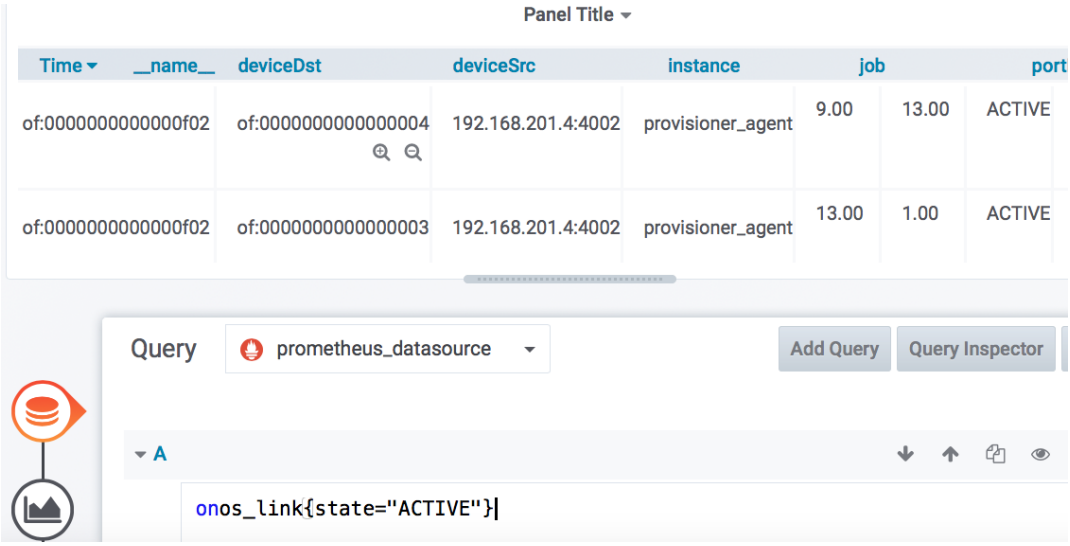

Figura 5.8: Selección de datos a representar

Los datos que hemos elegido se muestran en la siguiente tabla. Podemos comprobar que lo está haciendo de forma correcta, lo que significa que el sistema de monitorización está funcionando de forma correcta.

Recogemos los datos, los cuales se almacenan en Prometheus y posteriormente son visualizados en Grafana.

Capítulo 6

Conclusiones y trabajo futuro

6.1. Conclusiones

El trabajo realizado en este proyecto se enmarca dentro de las actividades de la iniciativa OnLife Networks de Telefónica, en el que se está desarrollando una arquitectura de Edge Computing basada en SDN. En este proyecto, y tal como se describe en el primer capítulo de este trabajo, se ha constatado que las redes actuales se encuentran en una situación de menor desarrollo que los sistemas, las cuales han evolucionado hacia configuraciones más flexibles, dinámicas y escalables. De esta manera, hoy en día, las redes no están preparadas para abordar la complejidad que implica satisfacer los nuevos servicios y aplicaciones demandados por los usuarios y soportar el uso que estas hacen de la red. Adicionalmente, las nuevas tecnologías aumentan la complejidad, ya que se despliegan y conviven con tecnologías anteriores e incluso legadas.

En este contexto, se ha verificado que SDN es la mejor opción para introducir la softwarización en las redes y poder dar respuesta a los nuevos servicios y aplicaciones, demandados por los usuarios. Por esta razón, en este trabajo se han analizado los diferentes protocolos de comunicación de SDN, identificando OpenFlow como un estándar *de facto* debido a la popularización de su uso. También se han analizado los distintos controladores SDN existentes, centrando el trabajo en ODL y ONOS, puesto que son los más implantados en múltiples entornos, identificando sus características, sus fortalezas y sus debilidades.

En este análisis se ha concluido que ONOS tiene un gran potencial para los operadores de telecomunicaciones, ya que mediante el proyecto CORD, se está impulsando un nuevo modelo redes donde la central telefónica se convierte en un centro de procesamiento de datos desde el que se pueden prestar nuevos servicios y aplicaciones.

Por otro lado, una conclusión clave de este trabajo es que el software se convierte en un elemento clave para hacer una realidad este nuevo modelos de redes, en el que existe una clara separación entre hardware y software y donde el hardware tiende a comoditizarse. Para desarrollar este software es necesaria una estrecha colaboración con las empresas fabricantes de hardware, pues aunque el objetivo es el de independizar el hardware totalmente del software, es algo que todavía queda lejos ya que todavía

existe una gran dependencia entre los fabricantes y los protocolos que estos usan. Por esta razón, desarrollar un software abierto que sea compatible con hardware de múltiples fabricantes es el objetivo a lograr, pero es algo que no se ha conseguido hasta el momento, por lo que es fundamental que iniciativas como OpenOLT alcancen sus objetivos.

Por último, respecto a la monitorización, es un tema que hasta la fecha no se ha tenido en mucha consideración. Es ahora cuando están surgiendo herramientas adecuadas para la creación y gestión de un sistema como tal. Tampoco se encuentran referencias para la implantación de un sistema de monitorización de red, por lo que se puede considerar que este campo será impulsado por los nuevos modelos de redes.

6.2. Trabajo futuro

En el presente proyecto se ha implementado una arquitectura de red actual con la que resolver los problemas derivados del gran aumento del tráfico de datos y del acceso a servicios remotos. Para ello, se ha debido realizar un estudio de todos los componentes que forman parte de esta arquitectura, los cuales aún requieren de mucho estudio y de la elaboración de más casos prácticos.

Los próximos puntos a desarrollar serían:

- Implementación de servicios: actualmente solo contamos en los nodos de computación con el software de ONOS y VOLTHA, con los que se ha podido probar la comunicación del tramo óptico. Es decir, la respuesta entre un cliente y un host. El objetivo es empezar a desarrollar servicios de red virtualizados (DHCP, NDP, VIdeo, etc.) dentro de los propios host.

- Integración completa de OpenOLT: se debe seguir estudiando la posibilidad de la integración del adaptador OpenOLT. Debido a la existencia de ciertos problemas como consecuencia de la falta de apoyo de terceros, no se ha podido profundizar en su uso como se hubiera querido. La solución a ello ha sido optar por las versiones de los adaptadores específicas para cada fabricante.

- Provisión automática: tanto la creación de EP como el registro de los dispositivos se ha realizado de forma manual. Sin embargo, se debe trabajar en un sistema que automatice todo el proceso.

- Elaboración de métricas: se ha profundizado poco en el sistema de monitorización construido y en las métricas que se obtienen del mismo. Un punto importante sería investigar otras herramientas para la construcción del sistema de monitorización.

- Creación de alertas: actualmente solo se están recogiendo y almacenando los datos de la monitorización. Es necesario la creación de alertas a partir de los datos obtenidos, e incluso, de elaboración de respuestas automáticas del sistema ante estas alertas. Por ejemplo, si no se detecta tráfico requerido en un flow, se debe alertar al administrador y seguidamente analizar los componentes que intervienen en ese flujo de tráfico.

Referencias

[1] *SERIES G: TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS Digital sections and digital line system – Optical line systems for local and access networks Gigabit-capable passive optical networks (G-PON): Transmission convergence layer specification.*

[2] *SERIES G: TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS Digital sections and digital line system – Optical line systems for local and access networks ONU management and control interface (OMCI) specification.*

[3] Caridad Anias Calderón-Frank Camilo Casmartiño Carlos Manuel Rodríguez Vergel, Alejandro García Centeno. Controladores sdn, elementos para su selección y evaluacion. jun 2014.

[4] http://blogtelecomunicaciones.ramonmillan.com/2008/05/el-protocolo omci.html. El protocolo omci, may 2008.

[5] https://blog.hostdime.com.co/edge-computing-que-es-para-que sirve/. Edge computing, que es y para que sirve, jun 2018.

[6] https://grafana.com/docs/. Grafana, jun 2018.

[7] https://howdoesinternetwork.com/2019/clos topology. topologia-clos, jun 2018.

[8] https://onosproject.org/board/. topologia-clos, jun 2018.

[9] https://prometheus.io/docs/introduction/overview/. Prometheus, jun 2018.

[10] https://searchdatacenter.techtarget.com/es/consejo/Cual-es-la-mejor-topologia-de-red-para-el-centro-de datos. topologias, jun 2018.

[11] https://searchdatacenter.techtarget.com/es/definicion/OpenFlow. Qué es openflow, nov 2012.

[12] https://unpocodejava.com/2019/01/23/que-es grpc/. Qué es grpc, jan 2019.

[13] https://wiki.onosproject.org/display/ONOS/System+Components. componentes-onos, jun 2018.

[14] https://wiki.opendaylight.org/view/Main$page.topologia - clos, jun$2018.

[15] https://www.adslzone.net/foro/fibra-optica.94/analisis-pon-que-es-olt-onu-ont odn.461996/. Analisis pon, oct 2018.

[16] https://www.business-solutions.telefonica.com/es/cloud-hub/knowledge-center/what-is-edge computing/. Edge computing, que es y para que sirve, jan 2019.

[17] https://www.ciena.com.mx/insights/what-is/What-is Network-Functions-Virtualization$_e s_L A.html.Quéesnfv, jun$2018.

[18] https://www.ciena.com.mx/insights/what-is/What-is SDN$_e s_L A.html.Quéessdn, jun$2018.

[19] https://www.opencompute.org/products/234/edgecore asxvolt16. Edgecore-olt, jun 2018.

[20] https://www.opendaylight.org/ecosystem-solutions/find-a solution. topologia-clos, jun 2018.

[21] https://www.opennetworking.org/voltha/. Voltha, jun 2018.

[22] http://tibitcom.com/architecture/. Tibit-olt, jun 2018.