UNIVERSIDAD TECNOLOGICA NACIONAL

TRABAJANDO CON
SDN Y OPENFLOW

Memorias de los trabajos realizados en el ámbito del proyecto de investigación UTN-2422 "Modelo para la evaluación de performance mediante identificación de tráfico y atributos críticos en Redes Definidas por Software

Autores: Reinaldo Scappini / Ricardo Calcagno / Facundo Alarcon / Sergio Gramajo / Diego Bolatti

Trabajando con SDN y OpenFlow : Memorias de los trabajos realizados en el ámbito del proyecto de investigación UTN-2422: modelo para la evaluación de performance mediante identificación de tráfico y atributos críticos en redes definidas por software / Reinaldo Scappini...[et al.]. - 1a ed compendiada. - Resistencia : Sergio Daniel Gramajo, 2019.

Libro digital, PDF

Archivo Digital: descarga y online
ISBN 978-987-86-1677-3

1. Redes de Información. 2. Protocolos de Comunicación. 3. Nuevas Tecnologías. I. Scappini, Reinaldo.
CDD 005.4

Prólogo

La constante evolución tecnológica de los últimos años implica también que cada vez sean más son los servicios y contenido que necesitan los usuarios de las telecomunicaciones. Su implementación, hoy por hoy, puede marcar si una región o país es desarrollado o en vías de ello. Es por ello que los organismos de estandarización internacionales están creando "nuevas maneras"de afrontar los desafíos de la gestión del desarrollo de infraestructuras de redes de información a gran escala e incrementar su productividad, adaptabilidad, flexibilidad, seguridad, multiplicidad de servicios, entre otras variables críticas e importantes.

A su vez, la combinación de los tipos de tráfico de red, servicios en ciudades y usuarios dinámicos necesita ser analizada considerando aspectos como variabilidad de las necesidades en el tiempo, los servicios en la nube, la virtualización de servidores y la necesidad de implantar adaptaciones ágiles según las demandas de performance de los usuarios, lo que adiciona complejidad a este escenario si trabajamos con redes creadas o estandarizadas hace décadas.

Para hacer frente a estos desafíos, la gestión de las redes está buscando responder con nuevas técnicas y tecnologías y cubrir sus necesidades actuales y futuras. En este libro hacemos una revisión completa, desde sus inicios, de una nueva tecnología que es tendencia en infraestructura de redes denominada Redes Definidas por Software (SDN). Revisamos, además, el protocolo OPENFLOW como base operativa de SDN.

Los contenidos que exponemos aquí presuponen un conocimiento previo del lector de diversos protocolos y estándares de las redes convencionales ya que se hace una comparación entre ambos paradigmas. No pretendemos, y esto puede ser una crítica, incluir temas de protocolos que se estudian en los cursos de grado de la Universidad, pero tratamos de que el compendio de revisiones esté lo suficientemente profundizado para su comprensión.

Además, los resultados que hemos obtenido se exponen en escenarios simulados con herramientas de Software Libre de modo que sea de utilidad para quienes quieren testear esta tecnología y aprender de sus ventajas o posibilidades en un futuro próximo.

Esta obra es el resultado de las memorias de los trabajos realizados en el ámbito del proyecto de investigación UTN-2422 “Modelo para la evaluación de performance mediante identificación de tráfico y atributos críticos en Redes Definidas por Software” de la Facultad Regional Resistencia de la UTN. Y fue desarrollado desde el año 2015 a 2018 en una línea de investigación creada para Redes de Información y en la que participaron expertos docentes-investigadores y alumnos de la casa de estudios.

Palabras clave : SDN; OPENFLOW; Simulación; Data Center

Dedicatoria

A nuestras familias...

Los autores

Agradecimientos

Nuestro gran agradecimiento y reconocimiento a los asesores científicos externos **PhD Luis Martínez López** y **PhD Macarena Espinilla Estévez**, ambos de la Universidad de Jaén, España.

Su colaboración en los inicios de nuestra línea de investigación ha sido crucial para lograr los objetivos alcanzados e introducir al equipo de profesores en metodologías de la investigación científica.

Su guía constante y profesionalismo han fortalecido a nuestro nuevo equipo en diferentes niveles colaborativos. Es por ello que queremos reiterar nuestro profundo agradecimiento a su asesoría.

Además queremos agradecer a los investigadores, especialmente al **Ing. Carlos Cuevas**, y becarios que participaron en diferentes momentos del proyecto y brindaron su aporte para lograr los resultados obtenidos.

Por último, a la diseñadora gráfica, **Carla Reinoso**, por su aporte y excelente trabajo en el diseño de la tapa del libro.

LOS AUTORES

Índice general

Índice de figuras

Autores y Menciones

Trabajando con SDN y OpenFlow son las memorias de los trabajos realizados en el ámbito del proyecto de investigación UTN-2422 : modelo para la evaluación de performance mediante identificación de tráfico y atributos críticos en redes definidas por softwarel. El mismo es un compendio del estado del arte para realizar los primeros pasos en SDN y es una investigación desde sus orígenes.

AUTORES Scappini, Reinaldo / Calcagno, Ricardo / Alarcón, Facundo / Gramajo, Sergio Daniel / Bolatti, Diego /

Resistencia, 20/08/2019

Capítulo 1

Evolución de las Redes Convencionales a Redes de Nueva Generación

Podríamos decir, como lo afirma Guy Pujolle en su libro Software Networks Virtualization, SDN, 5G and Security [17], que actualmente, la tecnología de redes está experimentando su tercera ola de evolución. La primera fue el cambio del modo de conmutación de circuitos al modo de intercambio de paquetes, la segunda, del modo cableado al modo inalámbrico. La tercera revolución, es el paso del hardware al modo de software. En este capítulo, para comenzar nuestro camino, y a modo de un resumido mapa de la travesía, empezaremos analizando los fundamentos de las redes convencionales originales y particularmente el ámbito donde inicialmente aparecieron las Redes Definidas por Software (Software Defined Networks o SDNs), este ámbito fue el de los Data Centers, enmarcados en la tecnología de los dispositivos y protocolos de redes "convencionales" de ese momento (que por lo demás y en gran medida, siguen completamente vigentes hoy en día); relevaremos la problemática que pretendieron resolver en esa instancia, y las vincularemos con la evolución de distintos conceptos de networking durante el mismo período; seguiremos analizando un conjunto de tecnologías que permitieron su desarrollo y adopción más allá de este entorno inicial (Virtualización de Funciones de Red, por ejemplo); para finalmente intentar una definición de las SDN, junto con sus componentes fundamentales y características esenciales.

1.1. Evolución de las redes - Conceptos y componentes de diseño

En la etapa inicial del estudio de SDN, creemos necesario repasar los conceptos subyacentes en el diseño y evolución de las redes y sus componentes. Como parte importante de nuestro trabajo en esta investigación, seleccionamos variedad de fuentes bibliográficas para los temas que desarrolla este libro. Es importante destacar que las fuentes y/o referencias bibliográficas están debida y rigurosamente explicitadas toda vez que se mencionan, aclarando además que aportan al ítem estudiado. En orden a esto, encontramos particularmente útil, por su claridad y enfoque, el libro "SDN: Software Defined

Networks" [1], cuyos autores son Thomas D. Nadeau y Ken Gray, su libro detalla muy bien la evolución conceptual en el diseño e implementación de las redes y para abordar los siguientes puntos nos fundamos, en gran parte, en ese contenido.

Comencemos por preguntarnos: ¿Cómo es una red típica? de, por ejemplo, un proveedor de servicios o de una organización, o bien un Sistema Autónomo (SA). Muy probablemente, al analizar y modelar una red de este tipo, nos encontraremos con una configuración similar a la de la Figura 1.1.

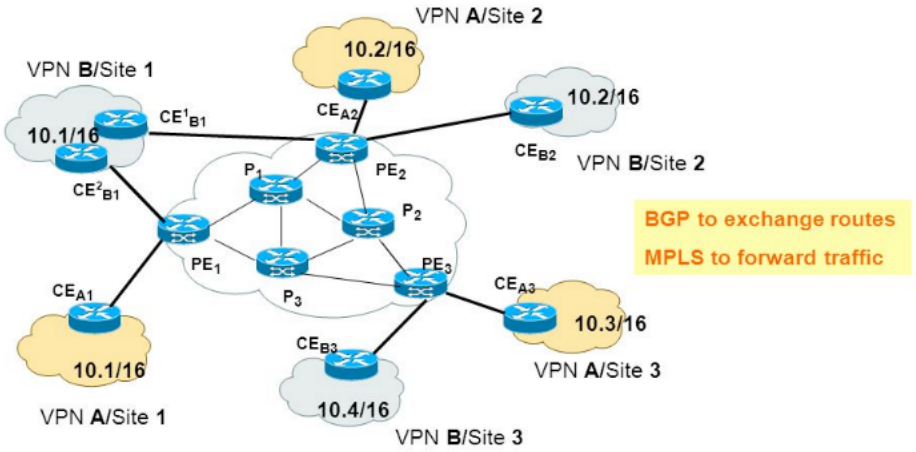

Figura 1.1: Esquema Genérico de la Red de un Sistema Autónomo (SA) u Organización

Este escenario consta de varios enrutadores, que corren algún protocolo de enrutamiento interno (típicamente OSPF [18]), organizado en áreas según la estructura interna de la organización. En muchos casos, si se trata de un Sistema Autónomo, tendrá relaciones de peering con otros SAs (Sistemas Autónomos), mediante el protocolo de enrutamiento exterior BGP (Border Gateway Protocol) [19].

En este caso, el administrador habrá configurado correctamente los anuncios de los prefijos de red y los vectores camino para los distintos SAs alcanzables a través de la organización. En algunos sitios finales, seguramente se contará con VPNs(redes privadas virtuales), para lo cual alguno de los enrutadores pueden ser precisamente VPN routers/firewalls; también probablemente, si las LANs en estos puntos finales son de cierta importancia, pueden estar implementadas como redes virtuales o VLANS; mientras que si el proveedor es de una envergadura importante, puede haber implementado un core (núcleo), con MPLS (MultiProtocol Label Switching) [20] para dar mayor capacidad de conmutación de tráfico, o bien dar soporte de VPNs mediante MPLS; en cuyo caso junto a los routers tradicionales, coexistirán algunos switches denominados multilayer.

En general podemos decir que estas redes están conformadas por dispositivos, genéricamente denominados dispositivos de networking, conectados entre sí. Para dejar un poco más completo nuestro panorama, supongamos que la organización utiliza seguramente algún tipo de aplicación basada en streaming (Skype, o similares), soporta audio o video en tiempo real (mediante cámaras y/o telefonía IP), tiene algún esquema más o menos elaborado de seguridad, integrando servicios de autenticación, encriptación (para las VPNs por ejemplo), nivel de acceso y permisos de usuarios, etc.; además de las consabidas aplicaciones típicas de una red corporativa (Servicio de Nombres, recursos compartidos, Active Directory [21]o similar, NFS (Network File System) [22], LDAP (Lightweight Directory Access Protocol) [23], etc.). Con esta descripción, muy probablemente estemos caracterizando una porción

muy importante de los SAs (Sistemas Autónomos), de tamaño pequeño/mediano que componen la Internet globalmente conectada.

La historia de Internet se corresponde aproximadamente con la evolución de los esquemas de control para gestionar la información de accesibilidad, los protocolos para la distribución de la información de accesibilidad y la generación algorítmica de rutas optimizadas frente a varios desafíos. En el caso de este último, esto incluye un crecimiento de la base de información utilizada (es decir, el crecimiento del tamaño de la tabla de rutas) y cómo gestionarla. No hacerlo puede dar lugar a la posibilidad de una gran inestabilidad en la red física. Esto, a su vez, puede llevar a altas tasas de cambio en la red o incluso a no funcionar. Otro desafío que se debe superar a medida que crece el tamaño de la información de enrutamiento es la difusión de la responsabilidad de la accesibilidad de la publicidad a partes de los datos de destino / destino, no solo entre las instancias locales del plano de datos sino también entre los límites administrativos.

Los Datacenters o centros de datos se diseñaron originalmente para separar físicamente los elementos informáticos tradicionales (por ejemplo, los servidores de PC), su almacenamiento asociado y las redes que los conectaban con los usuarios del cliente. La potencia de cómputo que existía en estos tipos de centros de datos se centró en la funcionalidad específica del servidor, ejecutando aplicaciones como servidores de correo, servidores de bases de datos u otra funcionalidad ampliamente utilizada para atender a los clientes de escritorio. Anteriormente, esas funciones, que se ejecutaban en los, a menudo, miles (o más) de escritorios dentro de una organización empresarial, estaban a cargo de servidores departamentales que proporcionaban servicios dedicados solo al uso local. A medida que pasaba el tiempo, los servidores departamentales migraron al centro de datos por una variedad de razones, en primer lugar, para facilitar la administración o control, y, en segundo lugar, para permitir el intercambio entre los usuarios de la empresa.

La separación de los planos de control y de datos es uno de los principios fundamentales de la SDN, y uno de sus más controvertidos también. Aunque no es un concepto nuevo, la forma de pensar contemporánea tiene algunos cambios interesantes en una idea antigua: a qué distancia se puede ubicar el plano de control del plano de datos, preguntas como:

- ¿Puede el 100 % del plano de control reubicarse más lejos que unos pocos centímetros?
- ¿Cuántas instancias se necesitan para existir para satisfacer los requisitos de flexibilidad y alta disponibilidad?
- ¿estrictamente centralizado?
- ¿semicentralizado o lógicamente centralizado?
- ¿totalmente descentralizado?
- La separación de los planos ¿tiene potencialmente ventajas económicas?

Las respuestas a estas y otras preguntas que surgen del debate sobre la separación de los Planos Control / Datos, son las premisas del desarrollo de la tecnología SDN. Respecto a esto, Thomas D. Nadeau y Ken Gray [1], manifiestan: La forma en que nos gusta abordar estas ideas es considerarlas como un continuo de posibilidades que se extiende entre el plano de control canónico completamente distribuido, el plano de control semi o lógicamente centralizado, y finalmente el plano de control estrictamente centralizado.

La Figura 1.2 ilustra el espectro de opciones disponibles para el operador de red, así como algunas de las ventajas y desventajas de cada enfoque.

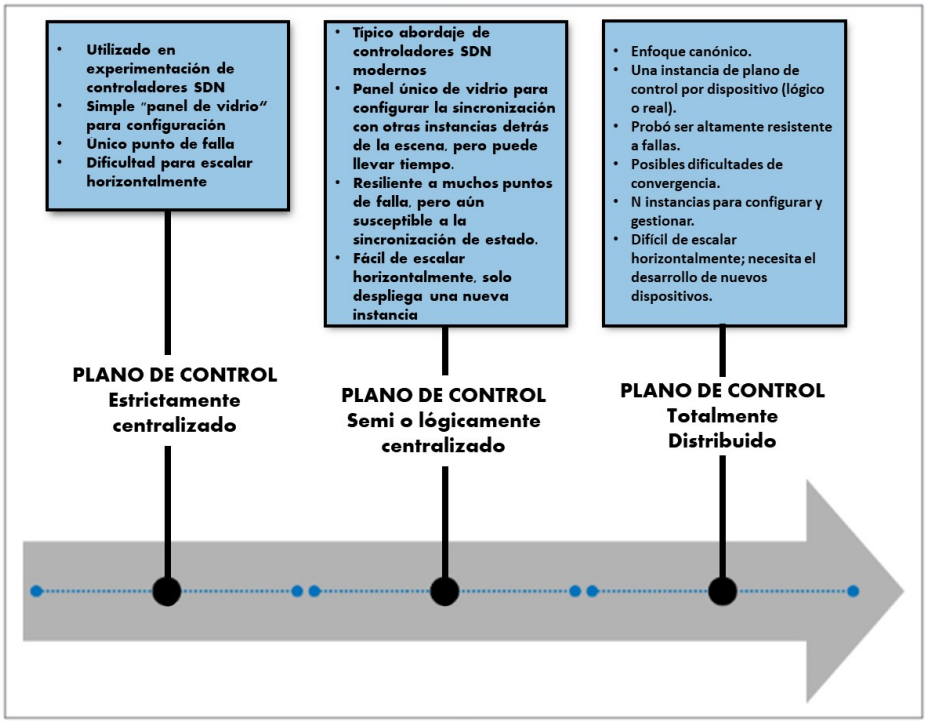

Figura 1.2: Espectro de control y opciones de distribución del plano de datos – Adaptado de [1]

Este es un abordaje a la temática SDN, que compartimos y con el que estamos totalmente de acuerdo.

1.2. Dispositivos de Networking

Centremos nuestra atención por un momento en los dispositivos que implementan las funciones mencionadas en el análisis de nuestra red tipo convencional, y que ejecutan los protocolos mencionados (OSPF, BGP, VPNs, VLANs, MPLS, etc.). Más allá de las obvias diferencias entre los equipos, dadas por los distintos fabricantes, e incluso por distintas líneas de productos dentro de un mismo proveedor, prácticamente todos ellos pueden representarse como "cajas" Figura 1.3, con múltiples interfaces, corriendo un software especializado (el IOS en caso de Cisco, por ejemplo) que permite la gestión del hardware específico (multichassis con procesadores, módulos de switching con varios puertos, interfaces específicas para algunos tipos de red como ATM [24], SMDS [25] enlaces seriales HDLC [26], etc.) que a su vez implementa las funciones de bajo nivel como clasificación y encolamiento de tramas/paquetes, buffers, funciones específicas como entramado o control de errores, o acceso múltiple compartido, gestión de enlaces inalámbricos, y demás. A un nivel más elevado, estas entidades elementales de routing/switching están encargadas de la gestión de la base de información de enrutamiento (RIB) o conmutación (FIB) y las funciones básicas de selección de rutas y conmutación.

En los siguientes apartados vamos a revisar los conceptos de las divisiones de planos, sus funciones y detalles de cada uno de ellos.

1.2.1. Mecánica de los planos de control y de datos en dispositivos de redes convencionales

Específicamente, la mecánica de los planos de control y de datos se muestra en la Figura 1.3, que representa una red de switches interconectados. En la parte superior de la figura, se muestra una red de switches, con una imagen expandida con los detalles de los planos de control y datos de dos de esos dispositivos (representados como A y B). En la 1.3, los paquetes son recibidos por el dispositivo A en el plano de control de la izquierda y, en última instancia, enviados al dispositivo B en el lado derecho de la figura.

Dentro de cada figura expandida, los planos de control y de datos están separados, con el plano de control ejecutándose en su propio procesador / tarjeta, y el plano de datos ejecutándose en otro plano separado. Ambos están contenidos dentro de un solo chasis. Discutiremos esta y otras variaciones sobre este tema de la ubicación física de los planos de control y datos más adelante en este capítulo. En la Figura 1.3, los paquetes se reciben en los puertos de entrada de la tarjeta de línea donde reside el plano de datos. Si, por ejemplo, se recibe un paquete que proviene de una dirección MAC desconocida, se envía o redirige (4) al plano de control del dispositivo, donde se aprende, se procesa y luego se reenvía. Este mismo tratamiento se da para controlar el tráfico, como los mensajes de protocolo de enrutamiento (por ejemplo, anuncios de estado de enlace OSPF). Una vez que se ha enviado un paquete al plano de control, la información contenida en el mismo se procesa y posiblemente se traduce en una alteración de la RIB, así como en la transmisión de mensajes adicionales a sus pares, alertándolos de esta actualización (es decir, una nueva ruta es aprendido). Cuando el RIB se vuelve estable, el FIB se actualiza tanto en el plano de control, como en el plano de datos. Posteriormente, el reenvío se actualizará y reflejará estos cambios. Sin embargo, en este caso, como el paquete recibido era uno de una dirección MAC no aprendida, el plano de control devuelve el paquete (C) al plano de datos (2), que reenvía el paquete en consecuencia (3). Si se necesita programación FIB adicional, esto también tiene lugar en el paso (C), que sería el caso por ahora que se ha aprendido la fuente de las direcciones

MAC.

El mismo algoritmo para el procesamiento de paquetes ocurre en el siguiente switch a la derecha.

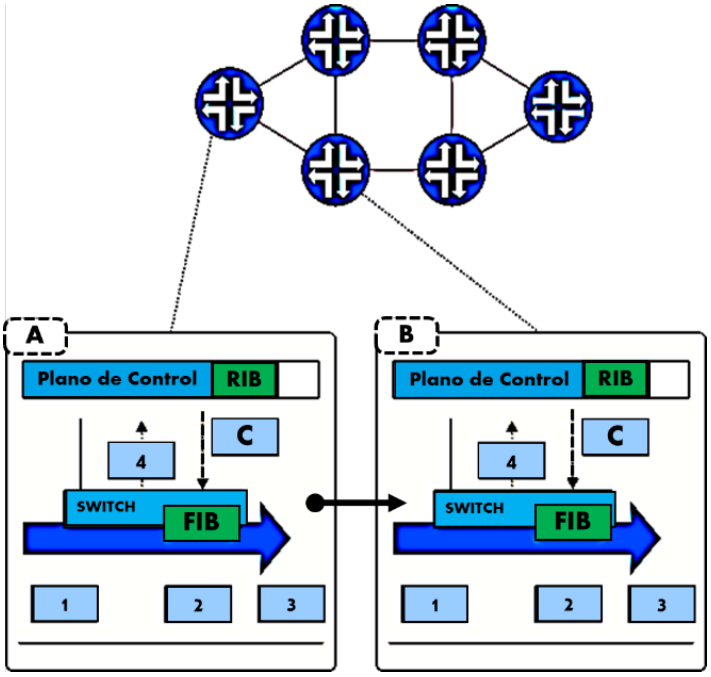

Figura 1.3: Planos de Control y Datos en Dispositivos de Networking Típicos- Adaptado de [1]

Todo este proceso, una vez más, controlado y gestionado por el software específico del fabricante, más una configuración específica elaborada y programada en el dispositivo por el administrador de la red.

1.2.2. Planos Operacionales

Analizamos aquí la gestión de los distintos Planos Operacionales que componen cada uno de estos dispositivos idealizados. Dentro del dispositivo, conviven varios PLANOS de Operación (Figura 1.4); típicamente, el plano de:

- DATOS
- CONTROL
- ADMINISTRACION

Algunos autores, incorporan también el plano de SERVICIOS.

1.2.3. Plano de Datos

Muy sucintamente, el plano de DATOS es el encargado de "mover" los datos del usuario dentro del dispositivo; aquí tenemos por ejemplo las funciones de clasificación y encolamiento, el movimiento "físico" de las tramas/paquetes desde la interface de entrada hacia la de salida; aquí son importantes las capacidades de transporte del backplane del dispositivo, con lo cual lo podemos asimilar con los buses de alta velocidad que comunican los componentes principales memoria, CPU, almacenamiento y slots de expansión de una computadora (Northbridge/Southbridge). Es también el componente que

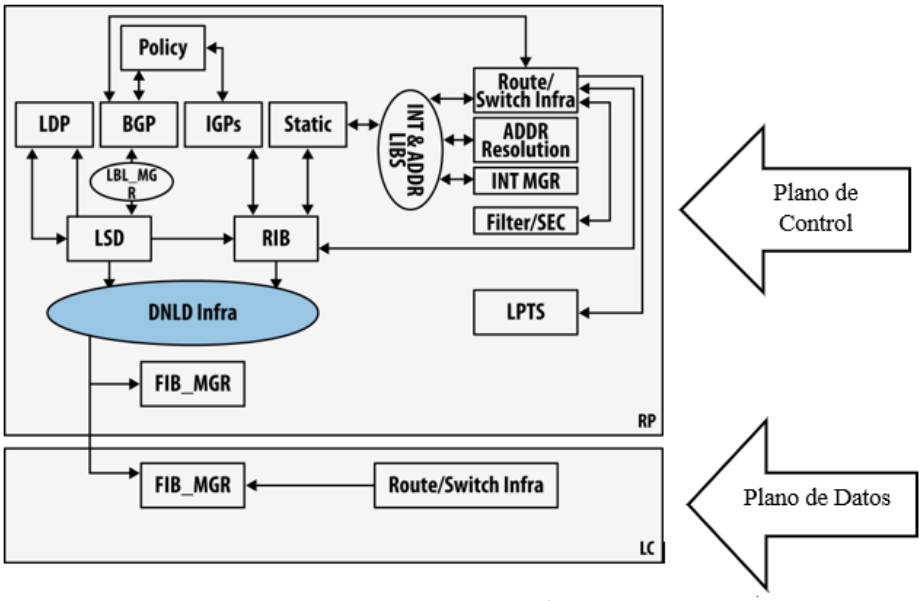

Figura 1.4: Ejemplo de Planos Operacionales- Tomado y adaptado de [1]

lleva un diseño más elaborado, de manera de hacer lo más eficiente posible el movimiento de los datos desde y hacia las interfaces y/o procesadores del dispositivo. También, cualquier modificación de los datos (cambio de prioridad, marcado DSCP, NAT, etc) se realiza en este plano. A veces se lo denomina Plano de Forwarding (Reenvío)

1.2.4. Plano de Control

El plano de Control es el que contiene la lógica para realizar las transferencias necesarias en el plano de Datos. Aquí "viven" las bases de información (LIB, MIB, RIB, etc.) para la toma de decisiones, como así también en este espacio se ejecutan los protocolos de enrutamiento que decidirán la mejor ruta para un paquete (OSPF, IGRP, BGP, MPLS...) como así también cualquier otra forma de procesamiento que haya definido el administrador, como las listas de acceso. Lo importante de recordar es que el plano de Control es el que decide de qué manera se mueven los paquetes/tramas en el plano de Datos.

1.2.5. Otros Planos

Los otros dos planos (Administración y Servicios) no son relevantes para este estudio, y se los menciona exclusivamente de modo enunciativo. Finalmente, mencionamos que en TODOS los dispositivos de networking conviven físicamente estos dos planos fundamentales que hemos analizado, el de Datos y el de Control. En realidad, el plano de control analizado es una combinación de planos de control de capa 2 o capa 3. Como tal, no debería sorprender, entonces, que se haya producido la misma progresión y evolución para las redes de capa 2 y capa 3 y los protocolos que conformaron estos planos de control. De hecho, la progresión de Internet ocurrió porque estos protocolos evolucionaron tanto en términos de funcionalidad como los proveedores de hardware aprendieron a implementarlos de maneras altamente escalables y disponibles.

1.2.6. Flujo de Datos en un Dispositivo Multislot

Un plano de control de capa 2 se centra en las direcciones de hardware o de capa física, como las direcciones IEEE MAC. Se construye un plano de control de capa 3 para facilitar las direcciones de capa de red, como las del protocolo IP. En una red de capa 2, los comportamientos relacionados con el aprendizaje de las direcciones MAC, los mecanismos utilizados para garantizar un gráfico acíclico (familiar para la mayoría de los lectores a través del Protocolo de árbol de expansión Spanning Tree [27]) y BUM (broadcast, unknown unicast, and multicast difusión) [28], crean su propio desafío de escalabilidad.

Ha habido varias iteraciones o generaciones de protocolos de control de nivel 2 basados en estándares cuyos objetivos fueron abordar éstas y otras cuestiones. En particular, estos incluyeron SPB / 802.1aq [29] del IEEE y TRILL [30] del IETF. Volveremos sobre esta última cuestión en el capítulo 2.

En una red de capa 3, el reenvío se centra en la accesibilidad de las direcciones de red. La información de accesibilidad de la red de nivel 3 se refiere principalmente a la accesibilidad de un prefijo de IP de destino. Esto incluye los prefijos de red en varias familias de direcciones para unicast y multicast. En todos los casos modernos, la red de capa 3 se utiliza para segmentar o unir dominios de capa 2 con el fin de superar los problemas de escalabilidad que presenta la capa 2 mencionados en el punto anterior. Específicamente, los puentes de la capa 2 que representan algunos conjuntos de subredes IP están típicamente conectados entre sí con un enrutador de la capa 3. Los enrutadores de capa 3 están conectados entre sí para formar redes más grandes, o rangos de direcciones de subred realmente diferentes. Las redes más grandes se conectan a otras redes a través de enrutadores de puerta de enlace que a menudo se especializan en la simple interconexión de redes grandes. Sin embargo, en todos estos casos, el enrutador enruta el tráfico entre las redes en la capa 3 y solo reenviará los paquetes en la capa 2 cuando sepa que el paquete ha llegado a la red de la capa 3 de destino final que luego debe entregarse a un host específico.

Una variante que desdibuja, o torna borrosa las fronteras capa2 / capa3 son los dispositivos LSR (Label Switching Router), que utiliza el protocolo MPLS (MultiProtocol Label Switching) [19] que es un estándar desarrollado por el IETF a partir de 1997. El protocolo MPLS, en realidad es un conjunto de protocolos, se formó sobre la base de combinar las mejores partes del reenvío (o conmutación) de la capa 2 con las mejores partes del enrutamiento IP de la capa 3 para formar una tecnología que comparte el reenvío de paquetes extremadamente rápido en forma similar a la que ATM (Asynchronous Transfer Mode) [24] implementó originalmente con técnicas de señalización de ruta muy flexibles y complejas adoptadas desde el mundo IP.

El protocolo EVPN [31] es un intento de resolver los problemas de escala de red de capa 2 que se describieron simplemente mediante el túnel de puentes de capa 2 distantes entre sí en una infraestructura MPLS o GRE (Generic Routing Encapsulation) [32]; solo entonces se intercambia información de alcance y direccionamiento de capa 2 a través de estos túneles y, por lo tanto, no contamina (ni afecta) la escala de las redes de capa 3 subyacentes. La información de accesibilidad entre puentes distantes se intercambia como datos dentro de una nueva familia de direcciones BGP, que de nuevo esto se hace sin afectar la red subyacente. El otro híbrido que vale la pena mencionar es LISP [33] el acrónimo LISP significa “Locator/ID Separation Protocol” [33] (ver también RFC 4984) [34]. En su esencia, LISP intenta resolver algunas de las deficiencias del modelo de plano de control distribuido general aplicado al multihoming [35], agregando nuevos dominios de direccionamiento y separando la dirección del sitio del proveedor en un nuevo protocolo de control y reenvío de encapsulación y mapa. En un

nivel ligeramente inferior, existen procesos de control complementarios particulares a ciertos tipos de redes que se utilizan para aumentar el conocimiento del plano de control mayor. Los servicios proporcionados por estos procesos incluyen la verificación / notificación de la disponibilidad de enlaces o información de calidad, descubrimiento de vecinos y resolución de direcciones. Debido a que algunos de estos servicios tienen bucles de rendimiento muy ajustados (para tiempos de detección de eventos cortos), son casi invariablemente locales al plano de datos (por ejemplo, OAM), independientemente de la estrategia elegida para el plano de control. Esto se muestra en la Figura 1.5 con los diversos protocolos de enrutamiento, así como el control de RIB a FIB que comprende el corazón del plano de control.

Tenga en cuenta que no estipulamos dónde residen los planos de control y de datos, sólo que el plano de datos reside en la tarjeta de línea (que se muestra en la Figura 1.5, en el cuadro de LC), y que el plano de control está situado en el procesador de ruta (indicado por la caja de RP).

Es interesante de analizar el flujo de los datos en un dispositivo Multislot, entendemos por tal, un dispositivo de networking (típicamente un router multichassis o un switch multilayer) (ver Figura 1.5) compuesto de un procesador central que comanda todo el dispositivo, más un conjunto de placas que a su vez, gestionan cada una de ellas un grupo de interfaces de un determinado tipo. Algunas pueden ser interfaces seriales de alta velocidad, otras pueden ser enlaces ATM, algunas más pueden ser interfaces Ethernet, etc. Este tipo de dispositivos permite una gran flexibilidad, ya que podemos agregar placas controladoras en función de nuestras necesidades.

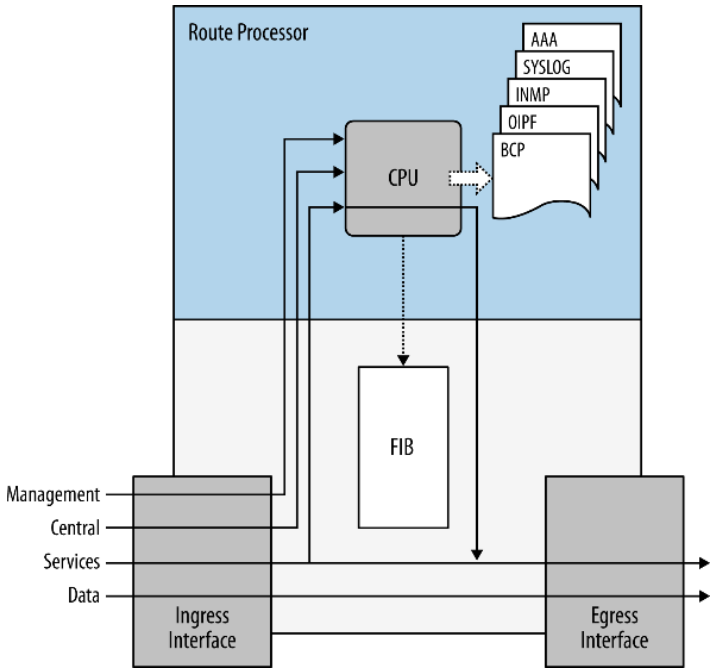

Figura 1.5: Ejemplo de Implementación Dispositivo de Networking Multislot – Tomado de [1]

En una arquitectura de este tipo, podemos verificar que el plano de control se encuentra ubicado en el módulo principal, junto con la CPU (y los planos de administración y servicios), en tanto que el plano de datos típicamente se ejecuta en cada uno de los módulos o tarjetas controladoras de las interfaces. De esta manera, podemos pensar que el plano de datos y el de control no residen "físicamente" en la misma ubicación; uno se encuentra en la interfaz y el otro en el módulo principal. O, lo que es más,

podemos pensar que ambos planos se encuentran desacoplados físicamente.

Este concepto nos será de suma utilidad cuando analicemos una característica fundamental de las redes SDN.

1.2.7. Plano de Control Hibrido

Para concluir este análisis, muy simplificado, de la arquitectura de los dispositivos de networking, consideremos un dispositivo idealizado, donde no todos los planos convivan en el mismo dispositivo. Es decir, llevemos un paso más adelante la idea del desacople de los planos, enunciada en el último párrafo del punto anterior.

Podemos concebir entonces un dispositivo, o mejor, un conjunto de dispositivos, donde en cada uno exista solamente el plano de datos (lo cual es obvio, ya que necesitamos mover los datos dentro del dispositivo), y donde el plano de control sea distribuido, esto es, una parte, mínima del control se realizaría en forma local al dispositivo, en tanto que el grueso de esta función estaría centralizada (junto con los planos de administración y de servicios) en una especie de Controlador, que supervisara dicha función de control para cada uno de los dispositivos conectados a él. Este tipo de arquitectura a veces se denomina con Plano de Control Híbrido, por obvias razones (ver Figura 1.6).

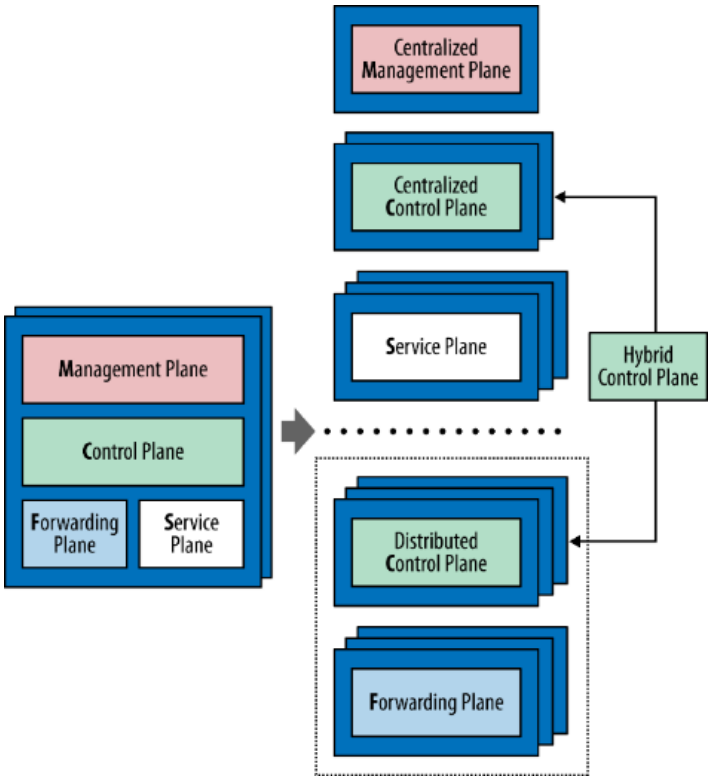

Figura 1.6: Plano de Control Hibrido – Tomado de [1]

1.3. Máquinas Virtuales: La Disrupción Conceptual

El esquema tradicional de los dispositivos de networking, someramente detallado en páginas anteriores, se vió cuestionado severamente con la aparición a gran escala de la virtualización de

Servidores y el surgimiento de Data Centers cada vez mayores y más complejos, que posibilitaron entre otras cosas la computación en la nube [36]. Para entender el problema o desafío al que nos enfrentamos, es imprescindible comprender el flujo de datos en este tipo de infraestructuras.

Normalmente un Data Center que alberga uno o varios servidores de virtualización (en adelante Hypervisor), que a su vez corren múltiples servidores virtualizados (por ejemplo, en forma de máquinas virtuales o VMs), están conectados al mundo exterior mediante switches, generalmente configurados con VLANs. A cada hypervisor que alberga las VMs se les asigna un puerto de VLAN, ya sea en forma estática o dinámica donde cada VM está conectada a una VLAN (A, B o C en color rojo, verde y naranja respectivamente), ver Figura 1.7.

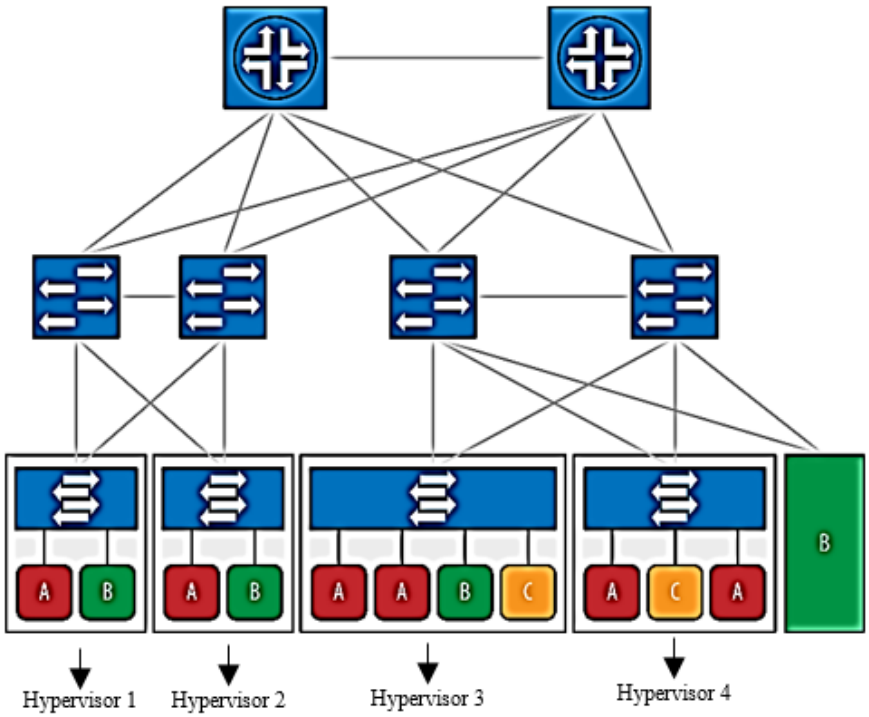

Figura 1.7: Esquema de un Data Center Multi-Propietario (Multi Tenant) – Adaptado de [1]

En esta Figura se muestran 4 hypervisores, que a su vez alojan MVs. El inconveniente se presenta cuando queremos mover una VM de un hypervisor a otro. Deberemos reasignar el nuevo puerto de switch para que participe de la VLAN a la que originalmente estaba asignado nuestro hypervisor anterior, o bien debemos cambiar de posición los cables (cambiar de puerto) en el switch para lograr el mismo cometido. En cualquier caso, el proceso no es automático, y puede ser aún más engorroso, al punto de volverse insoluble, si estamos en una configuración "multi tenant" (multi-propietario), como el de la Figura 1.7.

Un vistazo al gráfico proporcionado debería darnos un panorama del problema que se nos presenta cuando queremos organizar las VLANs en el Data Center para dar servicio a las VMs en los hypervisores. Por ejemplo, a qué VLAN debería pertenecer la interfaz del hypervisor 3, que tiene corriendo 4 VMs. Si queremos agregar una VM "C" en el primer hypervisor, ¿cómo deberíamos modificar la asignación de VLANs? ¿Debemos modificar también el cableado? El esquema de la conectividad a nivel capa 2, está graficado en la Figura 1.8.

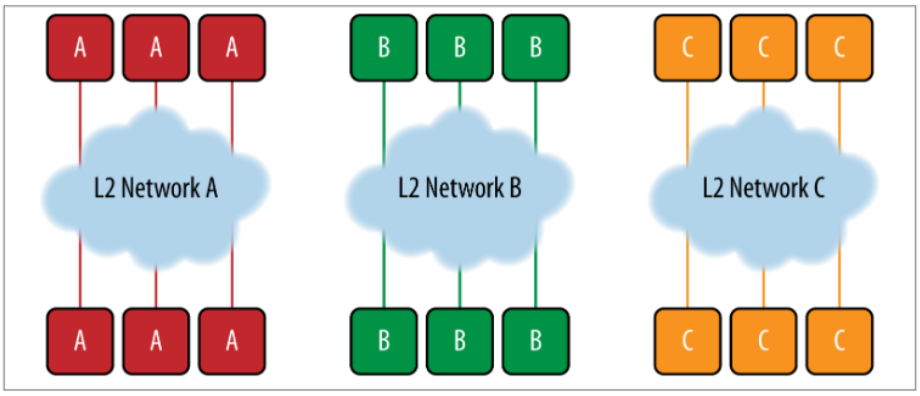

Figura 1.8: Conectividad desde el punto de vista lógico (VLANs) – Tomado de [1]

Lo expuesto en este ejemplo, con lo dicho en el punto anterior, debería servir para ilustrar la complejidad asociada a la administración de VMs en un Data Center, en su relación con VLANs y switches convencionales.

La solución a este inconveniente, y otros que por razones de espacio no comentamos aquí, llegó de la mano de Open vSwitch [37], que evolucionó posteriormente para convertirse en uno de los componentes clave de la arquitectura de las Redes Definidas por Software o SDN.

1.4. Open vSwitch

Cuando analizamos más en detalle el problema presentado anteriormente, nos damos cuenta de que falta un componente esencial en nuestra arquitectura. Si "virtualizamos" nuestros servidores y los convertimos en VMs, el inconveniente radica en que, para las funciones de red, utilizamos un switch "físico", con VLANs e interfaces reales. Sería mucho más conveniente contar con un switch "virtual" dentro del Hypervisor, ver Figura 1.9 que permita conectar todas las VMs y, a su vez, permita la conexión con una, o varias, interfaces físicas. Este switch virtual se administra y configura de manera estándar, es un componente del sistema operativo, y como tal sujeto a toda la administración de permisos, implementación de firewalls, y demás configuraciones asociadas.

Volviendo a nuestro problema original, cuando nosotros debemos mover una VM de un Hypervisor a otro, nos basta configurar las VLANs en el vSwitch en el nuevo Hypervisor para poder mantener la conectividad con el mundo exterior, sin alterar el cableado o la configuración VLAN de los switches físicos. Este proceso de reconfiguración del vSwitch puede hacerse en forma manual o automatizada, por ejemplo, mediante un script, al tratarse de un componente del sistema operativo.

Open vSwitch [37] surge como una evolución de este concepto inicial, y actualmente es un componente en sí mismo Figura 1.10

La aparición de Open vSwitch abrió las puertas a un nuevo concepto, el de Virtualización de las Funciones de Red (Network Function Virtualization - NFV). Entendiendo por tal, una abstracción de las funciones que realiza un dispositivo de red (una tarjeta NIC, un switch, un router) y su implementación como funciones de software, generalmente realizadas por un componente del Sistema Operativo hypervisor. Es evidente la similitud de este concepto, con el de virtualización de cualquier otro componente de un sistema de computación: disco, procesador, etc. NFV permite abstraer,

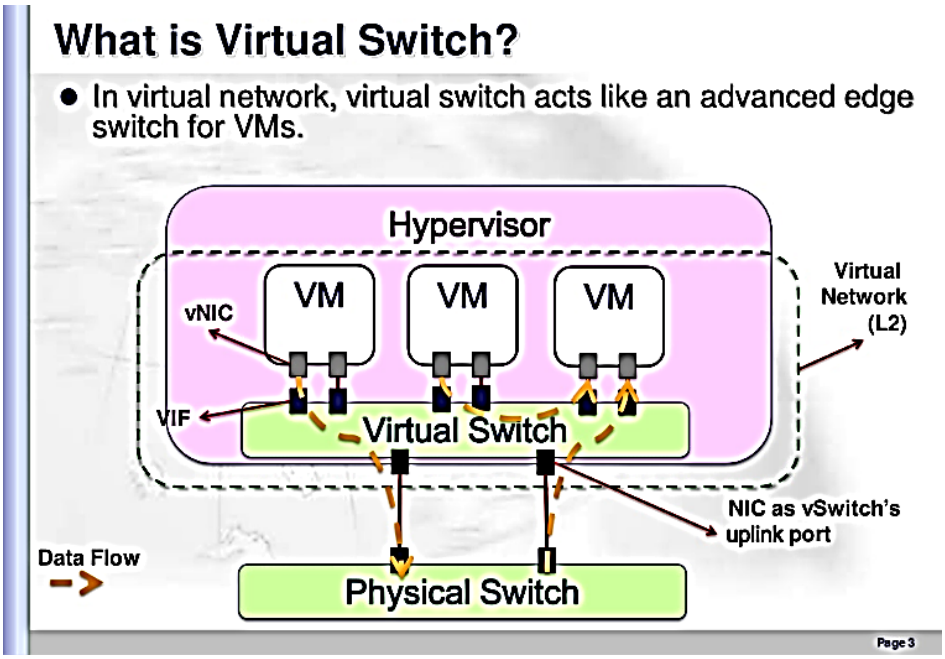

Figura 1.9: Switch Virtual –Tomado de [2]

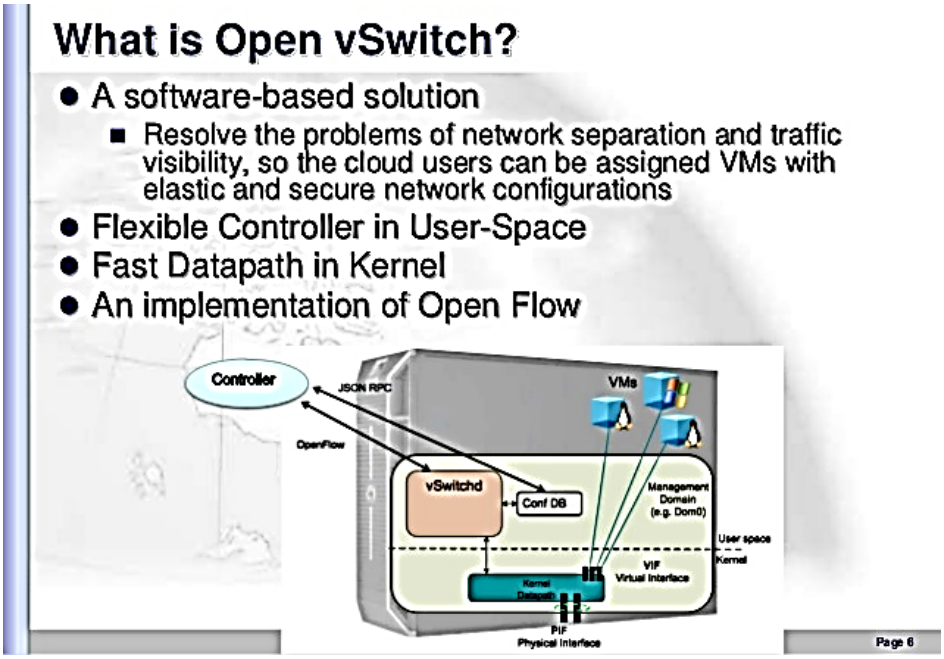

Figura 1.10: Open vSwitch – Tomado de [2]

modelizar e implementar los componentes de un dispositivo, particularmente las funciones de Plano de Control y Plano de Datos mencionadas oportunamente, con el atractivo de que dichas funciones, al estar virtualizadas, no Requieren de su Existencia Física (Ejecución) en el mismo dispositivo. Es perfectamente concebible un dispositivo de networking, virtualizado, donde el plano de Control y el plano de Datos se ejecuten como entidades distintas y potencialmente separadas, en dispositivos físicos o virtuales arbitrarios ver Figura 1.11. Este concepto probará ser muy poderoso, y un elemento crucial en la implementación de las SDN.

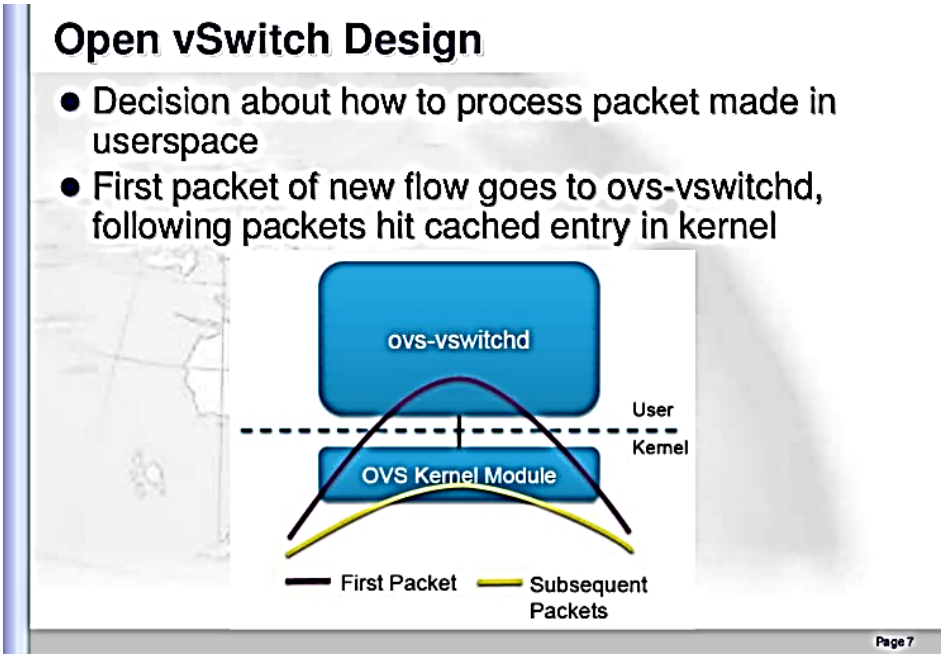

Figura 1.11: Procesos ejecutados en distintos espacios – Tomado de [2]

Como un ejemplo de esta separación o desacople de los planos mediante la NFV, y su relación con Open vSwitch, consideremos de qué manera se realiza el procesamiento de los paquetes en este componente. La función (o plano) de Control está implementada mediante el proceso ovs-vswitchd [38], un demonio que administra y controla cualquier número de switches Open vSwitch en la máquina local, y es un proceso de espacio de Usuario. Este es el responsable de decidir qué hacer con cada paquete que entra por una determinada interfaz virtual: reenviarlo, descartarlo, remarcarlo, etc. Esta decisión es comunicada a la función (o plano) de Datos, que reside en el Kernel (probablemente en el espacio del hypervisor). Este es el proceso encargado de físicamente mover los datos de una interfaz virtual a otra y/o de allí hacia las interfaces físicas administradas por el hypervisor. Como una optimización, el Plano de Datos puede guardar en memoria (cache) la decisión tomada para el primer paquete de un flujo, y repetirla para todos los siguientes del mismo flujo, ver Figura 1.12

Una mirada final sobre Open vSwitch, nos permitirá apreciar la flexibilidad (y complejidad) de este componente de software. Se puede integrar perfectamente con el resto de los componentes del SO hypervisor, y está diseñado para permitir la automatización prácticamente completa de la función de red, mediante extensión programática (se puede configurar por ejemplo mediante scripts), a la vez que soporta protocolos e interfaces de administración estándares (NetFlow [39], sFlow [40], IPFIX [41], RSPAN [42], CLI [43], LACP [44], 802.1ag [45]), y soporta distribución a través de varios servidores

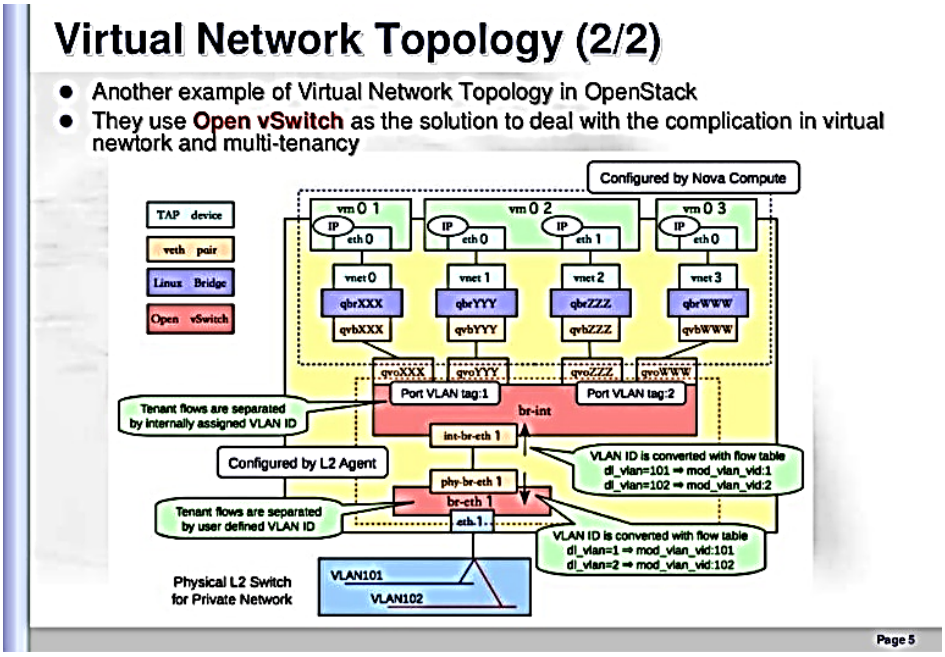

Figura 1.12: Ejemplo de implementación de Open vSwitch como componente de OpenStack [3] – Tomado de [2]

físicos. De todos modos, nuestro interés en este componente se basa en dos de sus características: nos permite ilustrar el concepto de separación de los planos de Control y Datos (concepto fundamental de SDN); y soporta OpenFlow, el otro componente esencial de SDN, del cual nos ocupamos más adelante en el punto 1.5

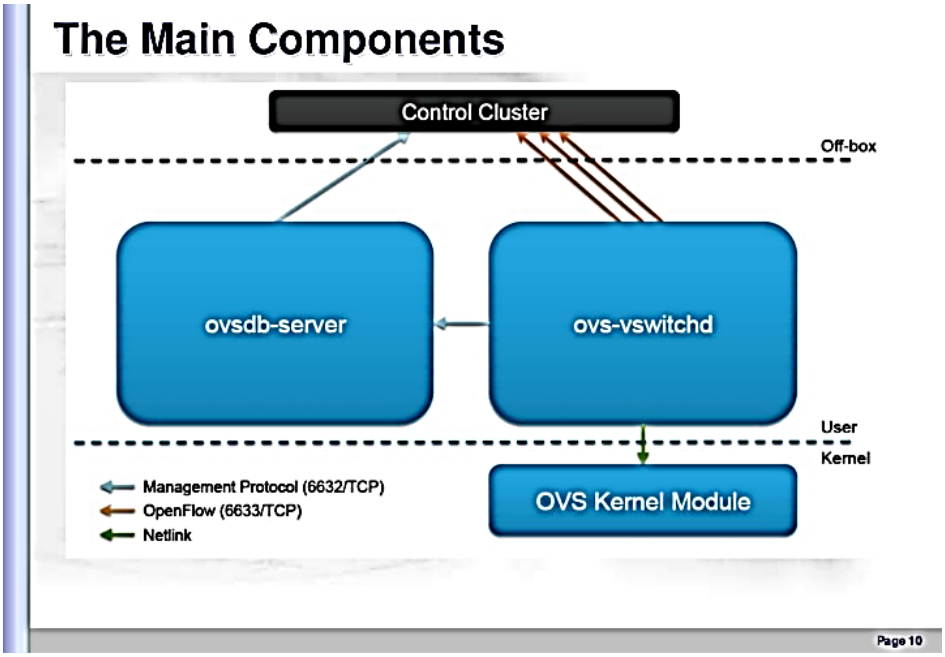

Figura 1.13: Operación en modo autónomo o coordinado mediante controlador – Tomado de [2]

Es oportuno mencionar otra característica de Open vSwitch: puede operar tanto en modo autónomo

(stand-alone), o coordinado junto a otros Open vSwitches, mediante el protocolo OpenFlow [46]. Esta última característica es de vital importancia para nosotros, ya que nos permitirá, por ejemplo, delegar toda la función del Plano de Control en una entidad centralizada, que coordine varios dispositivos pertenecientes a una topología, ver Figura 1.13. Es obvio que en este esquema necesitamos un canal de comunicaciones entre la entidad centralizada (en adelante, el Controlador) y los dispositivos administrados.

Para no dejar en abstracto el concepto de Open vSwitch, y para ilustrar su funcionamiento, mostramos un ejemplo de configuración de políticas de Calidad de Servicio (QoS) en un switch virtual. nótese que los comandos y opciones son mayormente auto documentados: es fácil deducir lo que hace cada uno, no obstante, cabe aclarar que existe extensa documentación al respecto [47] (ver Figura 1.14).

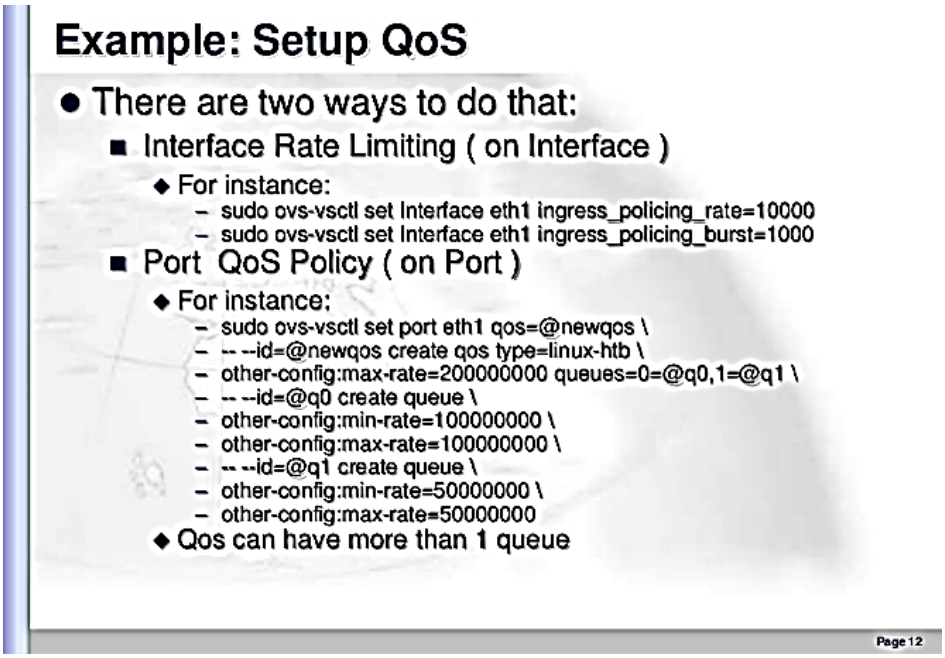

Figura 1.14: Configurando parámetros de calidad de servicio en un Open vSwitch – Tomado de [2]

Finalmente, mencionamos que estos comandos de configuración se utilizan en el modo autónomo; para el caso de la administración remota se utiliza el protocolo OpenFlowTM [46]. Analizaremos sus características a continuación.

1.5. Nacimiento de OpenFlow

La idea de redes programables y el plano de control desacoplado (lógica de control) del plano de datos existe desde hace muchos años, la podemos situar a mediados de la década de 1990. El Grupo de trabajo de señalización abierta (OPENSIG) inició una serie de talleres en 1995 para hacer que las redes de cajeros automáticos, Internet y móviles sean más abiertas, extensibles y programables. Motivado por estas ideas, un grupo de trabajo del Grupo de trabajo de ingeniería de Internet (IETF, por sus siglas en inglés) creó el Protocolo general de administración de switches (GSMP) para controlar un switch de etiquetas. Este grupo concluyó oficialmente y GSMPv3 [48] se publicó en junio de 2002.

Desde entonces, la iniciativa de red Active Network, propuso la idea de una infraestructura de red programable para servicios personalizados, sin embargo, Active Network no pudo conseguir avanzar, principalmente, debido a preocupaciones prácticas de seguridad y rendimiento. A partir de 2004, el proyecto 4D: Clean Slate Architectures for Network Management [49], abogó por un diseño limpio que enfatizara la separación entre la lógica de decisión de enrutamiento y los protocolos que rigen la interacción entre los elementos de la red. Las ideas en el proyecto 4D proporcionaron inspiración directa para trabajos posteriores como NOX [50], que proponía un sistema operativo para redes en el contexto de una red habilitada para OpenFlow. Más tarde, en 2006, el grupo de trabajo IETF Network Configuration Protocol propuso NETCONF [51], como un protocolo de administración para modificar la configuración de los dispositivos de red. El grupo de trabajo está actualmente activo y la última norma propuesta se publicó en junio de 2011.

El grupo de trabajo IETF Forwarding and Control Element Separation (ForCES) [52] está liderando un enfoque paralelo a SDN. SDN y Open Networking Foundation comparten algunos objetivos comunes con ForCES. En ForCES, la arquitectura interna del dispositivo de red se redefine a medida que el elemento de control se separa del elemento de reenvío, pero la entidad combinada aún se representa como un elemento de red único para el mundo exterior. El antecesor inmediato de OpenFlow fue el proyecto Ethane [53] de Stanford, que en 2006 definió una nueva arquitectura de red para redes empresariales. El objetivo de Ethane era utilizar un controlador centralizado para administrar las políticas y la seguridad en una red (ver Figura 1.15).

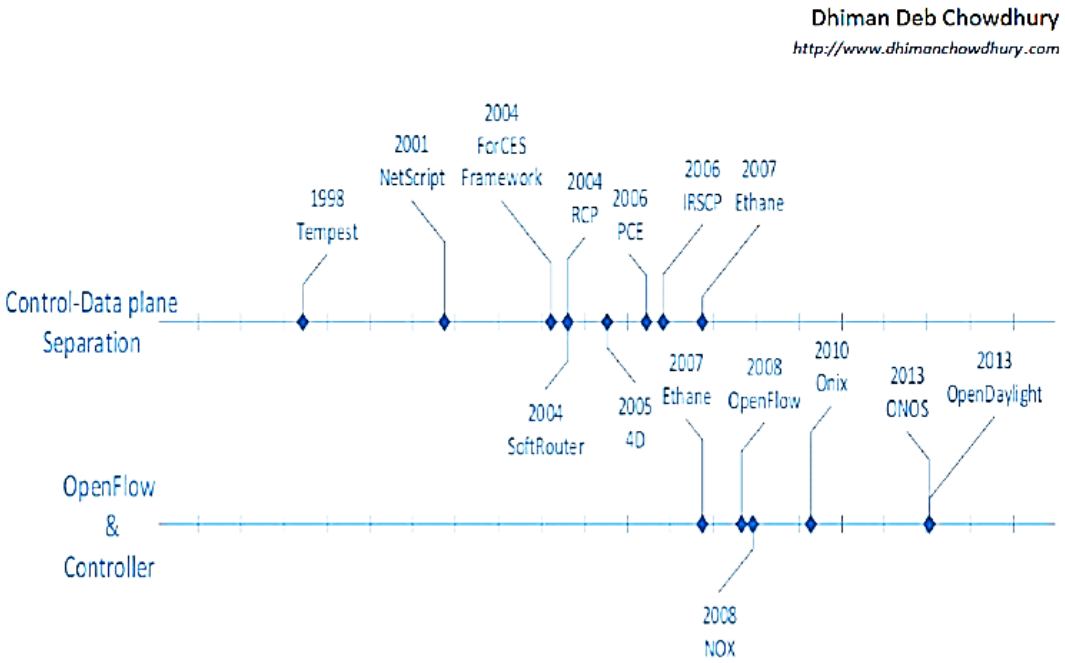

Figura 1.15: Línea de tiempo que representa el significado histórico de ForCES – Tomado de [4]

1.5.1. Génesis de OpenFlow

En 2007, un grupo de trabajo de la Universidad de Standford formado por los profesores Nick McKeown, Scott Shenker y el estudiante de doctorado Martín Casado desarrollaron OpenFlow y

fundaron Nicira, una compañía de virtualización de redes. Es a partir de este momento donde se sitúa el nacimiento de SDN.

En el año 2009 la U. de Stanford publica el estándar V 1.0 de OpenFlow [54]. En el año 2011 Scott Shenker y Nick McKeown fundaron la Open Networking Foundation (ONF), y se comienza a implementar en redes de producción. ONF es una organización que buscaba fomentar el uso de OpenFlow, la creación de estándares y la implantación de SDN más allá de las universidades, tal es así que en abril de 2012 publica un documento "Software-Defined Networking:The New Norm for Networks" [55], estableciendo los fundamentos de SDN. En julio de 2012 VMware, una de las compañías líderes en virtualización de servidores, dió un paso hacia la incorporación de la virtualización de redes en su gama de productos y compró Nicira por 1260 millones de dólares. Martín Casado pasó a ser el .ªrquitect networking.en jefe de VMware.

La idea germinal [56] de OpenFlow, fue lanzada inicialmente por el Programa Clean Slate [49], dirigido por Nick McKeown en la Universidad de Stanford [57]. Los coordinadores del programa identificaron cinco áreas clave para la investigación:

- Arquitectura de Red
- Aplicaciones Heterogéneas
- Tecnologías de Capa Física Heterogéneas
- Seguridad
- Economía y Políticas de Gestión

El programa Clean Slate se suspendió en enero de 2012, después de generar cuatro proyectos de seguimiento principales.

- Infraestructura de Internet: OpenFlow y redes definidas por software (SDN)
- Internet móvil: POMI 2020
- Redes Sociales Móviles: mobiSocial
- Centro de Datos: Laboratorio Experimental del Centro de Datos de Stanford

1.5.2. La Open Networking Foundation (ONF)

Es un consorcio de la industria sin fines de lucro que está liderando el avance de SDN y la estandarización de elementos críticos de la arquitectura SDN (ver [8]) tales como el protocolo OpenFlow ™, que estructura la comunicación entre los planos de control y planos de datos de dispositivos de red soportados. OpenFlow es la primera Interfaz estándar diseñada específicamente para SDN, que proporciona alto rendimiento y control de tráfico granular a través de dispositivos de red de múltiples proveedores. De esta manera, los switches delegan la función del Plano de Control en el controlador centralizado, mientras implementan las funciones del Plano de Datos, generalmente mediante hardware especializado. El dispositivo puede ser configurado y controlado remotamente mediante OpenFlow, principalmente en lo referente a las decisiones de conmutación y la base de información para conmutación.

En la actualidad, la especificación de OpenFlow, es mantenida, definida y actualizada por la Open Networking Foundation [58], se trata de una especificación por la que se establecen las funciones y los protocolos utilizados para la administración centralizada de switches mediante un controlador. OpenFlow es un protocolo de comando y control que incluye comunicación mediante canales seguros SSL/TLS, con funcionalidades tales como descubrimiento de características y configuración de los

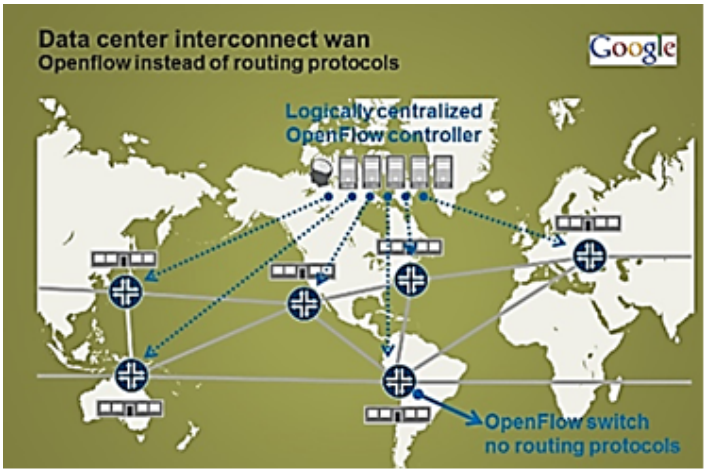

Figura 1.16: Ejemplos de implementación SDN

dispositivos por parte del controlador, como así también la manipulación de las tablas de reenvío en los dispositivos (switches).

Las SDN basadas en OpenFlow se están implementando actualmente en una variedad de dispositivos de red y software, que proporcionan beneficios sustanciales tanto para empresas como transportistas (carriers) Figura 1.16. Como muestra la Figura 1.17, ya en el año 2013 existía variedad de fabricantes que implementaban soporte OpenFlow.

Manufacturer	Switch models	OpenFlow version
Brocade	NetIron CES 2000 Series, CER 2000,	1.0
Hewlett Packard	3500/3500yl, 5400zl,6200zl, 6600, 8200zl	1.0
IBM	RackSwitch G8264, G8264T	1.0
Juniper	EX9200 Programmable switch	1.0
NEC	PF5240, PF5820	1.0
Pica8	P-3290, P-3295, P-3780, P3920	1.2
Pronto	3290 and 3780	1.0
Broadcom	BCM56846	1.0
Extreme Networks	BlackDiamond 8K, Summit X440, X460, X480	1.0
Netgear	GSM7352Sv2	1.0
Arista	7150, 7500, 7050 series	1.0

Figura 1.17: Lista de fabricantes y productos con implementación OpenFlow – Tomado de [5]

Básicamente OpenFlow define dos componentes: el componente de comunicación (formatos, mensajes, campos, protocolo de comunicación, etc. denominado Wire Protocol; y un conjunto de comandos de configuración agrupados en la especificación OF-CONFIG [6]. Un punto de configuración de OpenFlow se comunica con un contexto operacional que es capaz de soportar un switch de OpenFlow utilizando el Protocolo de configuración y administración de OpenFlow (OF-CONFIG) (ver Figura 1.18).

OF-CONFIG define un switch OpenFlow Figura 1.19, como una abstracción llamada OpenFlow Logical Switch. El protocolo OF-CONFIG permite la configuración de artefactos esenciales de un switch lógico OpenFlow para que un controlador OpenFlow pueda comunicar y controlar el switch lógico OpenFlow a través del protocolo OpenFlow.

El servicio que envía mensajes OF-CONFIG a un switch compatible con OpenFlow se denomina OpenFlow Configuration Point. No se hacen suposiciones sobre la naturaleza del punto de configuración de OpenFlow. Por ejemplo, puede ser proporcionado por un software que actúa como un controlador OpenFlow o puede ser proporcionado por un servicio proporcionado por un marco de administración de red tradicional.

En algunos contextos de despliegue, el OpenFlow Configuration Point y el controlador OpenFlow pueden pertenecer a diferentes entidades administrativas, por ejemplo, proveedor y cliente, respectivamente. Las interacciones entre los puntos de configuración de OpenFlow y los controladores de OpenFlow (Figura 1.20) están fuera del alcance de OF-CONFIG 1.2, pero se espera que se aborden en

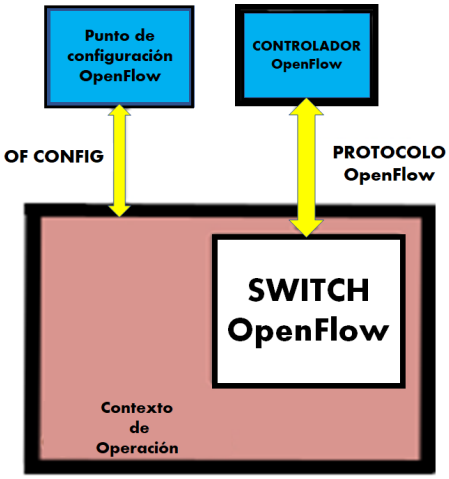

Figura 1.18: Configuración de OpenFlow – Adaptado de [6]

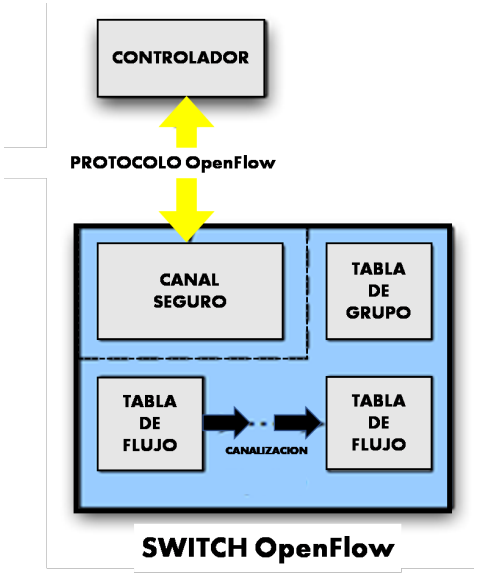

Figura 1.19: Switch OpenFlow – Adaptado de [7]

futuras versiones de la especificación.

OF-CONFIG 1.2 se centra en las siguientes funciones necesarias para configurar un switch lógico OpenFlow:

- La asignación de uno o más controladores OpenFlow a los planos de datos de OpenFlow
- La configuración de colas y puertos.
- La capacidad de cambiar de forma remota algunos aspectos de los puertos (por ejemplo, p / abajo)
- Configuración de certificados para la comunicación segura entre los switches lógicos OpenFlow y los controladores OpenFlow
- Descubrimiento de capacidades de un switch lógico OpenFlow
- Configuración de un conjunto de tipos de túneles específicos como IP-in-GRE, NV-GRE, VxLAN

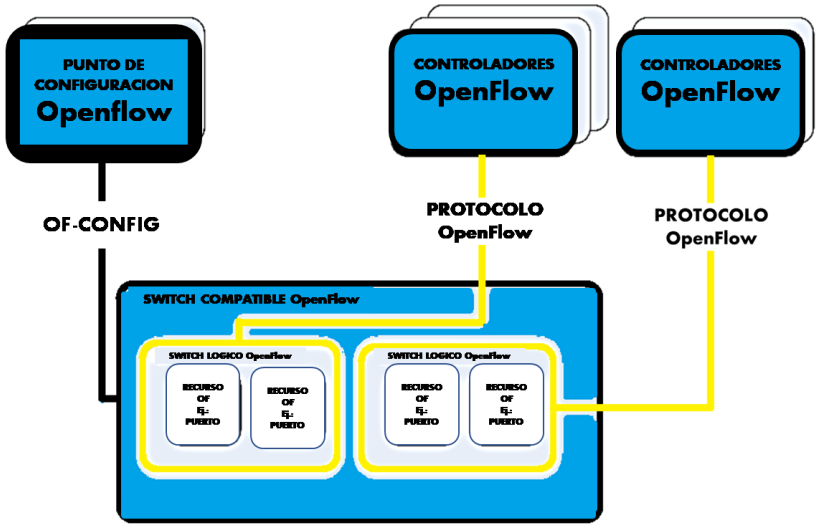

Figura 1.20: Relación entre los componentes definidos en esta especificación, el protocolo OF-CONFIG y el protocolo OpenFlow – Adaptado de [6]

1.5.3. Concluyendo

Podemos describir una SDN de la siguiente forma [59]:

La red definida por software aplica la flexibilidad del software a la totalidad del espacio de red.

Incluye un número ilimitado de relaciones comerciales, que abarcan geográficamente todo el mundo, desde la negociación y entrega del servicio al usuario final, hasta la planificación, instalación, aprovisionamiento y mantenimiento de la infraestructura de red. Además de reenviar el tráfico, una SDN puede procesar el tráfico, ya sea como parte de servicios de valor agregado, o para propósitos de servicio y mantenimiento de la red.

La arquitectura general está destinada a abarcar la totalidad del espacio. Sin embargo, está estructurado para facilitar subdivisiones, por ejemplo, en entornos de operador, nube, campus o empresa que compran gran parte de su servicio de red a terceros, a través de límites de dominio empresarial opaco o semiopaco.

SDN se basa en tres principios:

- Desacoplamiento del tráfico y el procesamiento de control
- Control lógicamente centralizado.
- Programabilidad de los servicios de red.

La definición formal de SDN brindada por la Open Networking Foundation se puede ver en [60] Citando a Guy Pujolle [17], “En conclusión, el mundo de las redes está cambiando enormemente, por las razones mencionadas anteriormente. Está cambiando más rápidamente de lo que se podría haber esperado hace unos años. Se presentó una proposición inicial, pero fracasó: comenzar de nuevo desde cero. Esto se conoce como el .enfoque tardío limpio": eliminar todo y comenzar de nuevo desde la nada. Lamentablemente, no se ha adoptado ninguna propuesta concreta y la transferencia de paquetes IP sigue siendo la solución para el transporte de datos. Sin embargo, en las numerosas proposiciones, la virtualización y la nube son las dos vías principales que se utilizan ampliamente en la actualidad.”.

La ONF, cuenta en la actualidad con muy importantes socios actores de la industria, además de colaboradores y desarrolladores (ver [61]). Múltiples fabricantes brindan soporte a OpenFlow en sus dispositivos de networking, como así también empresas internacionales de primer nivel tienen su infraestructura, o parte de ella gestionada con SDN. En el siguiente capítulo, nos enfocamos en los detalles de la arquitectura SDN y el protocolo OpenFlow.

Capítulo 2

Arquitectura SDN y Protocolo OpenFlow

En este capítulo examinamos en detalles los aspectos técnicos de la arquitectura de los protocolos SDN y OPENFLOW y su funcionamiento en conjunto.

En los siguientes apartados revisamos la arquitectura estándar, los planos operativos, la arquitectura de red y transporte SDN, además, las especificaciones del protocolo OPENFLOW.

2.1. Arquitectura SDN

La Open Networking Foundation (ONF), en un documento técnico [8], define y explica los componentes arquitectónicos que se ven en la Figura 2.1. ONF actualiza y evoluciona constantemente la terminología, que se rastrea en el proyecto Glosario de ONF.

2.1.1. Aplicación SDN

Las aplicaciones SDN son programas que comunican explícita, directa y programáticamente sus requisitos de red y el comportamiento deseado de la red a la SDN. Controlador a través de NBIs (. Además, pueden consumir una vista abstracta de la red para sus decisiones internas. Una aplicación SDN consta de una lógica de aplicación SDN y uno o más controladores NBI. Las aplicaciones SDN pueden exponer otra capa de control de red abstraído, ofreciendo así uno o más NBI (s) de nivel superior a través del agente (s) de NBI respectivo (no se muestra en la Figura 2.1).

2.1.2. Controlador SDN

El controlador SDN es una entidad lógicamente centralizada que se encarga de: (i) traducir los requisitos de la capa de aplicación SDN a las rutas de datos SDN y (ii) proporcionar a las aplicaciones SDN una vista abstracta de la red (que puede incluir estadísticas) y eventos). Un controlador SDN consta de uno o más agentes NBI, la lógica de control SDN y el controlador CDPI. La definición como una entidad lógicamente centralizada no prescribe ni excluye detalles de implementación tales como

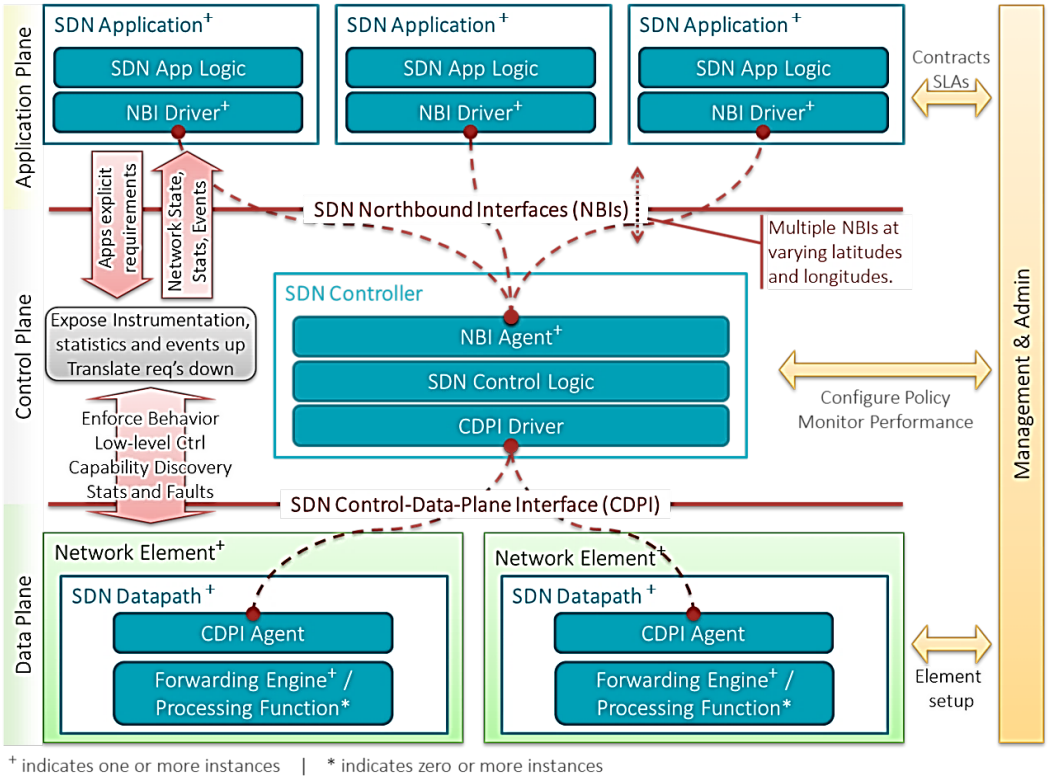

Figura 2.1: Visión general de -Arquitectura de Red Definida por Software tomado de [8]

la federación de múltiples controladores, la conexión jerárquica de los controladores, las interfaces de comunicación entre los controladores, ni la virtualización o el corte de los recursos de red.

2.1.3. SDN datapath

La implementación de referencia del switch OpenFlow de la Universidad de Stanford incluye “ofdatapath", que implementa la tabla de flujo en el espacio del usuario; “ofprotocol", un programa que implementa el componente de canal seguro del switch de referencia, y “dpctl", que es una herramienta para configurar el switch. Esta distribución incluye también algún software adicional (por ejemplo, un controlador, que consiste em un programa de controlador simple que se conecta a cualquier número de switches OpenFlow y un disector de Wireshark que puede decodificar el protocolo OpenFlow). El datapath, es un dispositivo de red lógico, que expone la visibilidad y el control del contenido de sus capacidades de procesamiento de datos y reenvío anunciadas. La representación lógica puede abarcar todos o un subconjunto de los recursos del sustrato físico. Un datapath comprende un agente CDPI y un conjunto de uno o más motores de reenvío de tráfico y cero o más funciones de procesamiento de tráfico. Estos motores y las funciones pueden incluir un simple reenvío entre las interfaces externas de la ruta de datos o las funciones internas de procesamiento o terminación del tráfico. Uno o más datapaths pueden estar contenidos en un único elemento de red (físico) una combinación física integrada de recursos de comunicaciones, administrada como una unidad. Un datapath SDN también se puede definir a través de múltiples elementos físicos de la red. Esta definición lógica no prescribe ni excluye los detalles de la implementación, como la asignación lógica a física, la gestión de los recursos físicos

compartidos, la virtualización o la división de la ruta de datos SDN, la interoperabilidad con redes que no son SDN, ni la funcionalidad de procesamiento de datos, que puede incluir funciones L4-7 (capas 4 a 7 del modelo OSI).

2.1.4. Control de SDN

Para la interfaz del plano de datos (CDPI): el CDN de la SDN es la interfaz definida entre un controlador SDN y un Datapath SDN, que proporciona al menos: (i) control programático de todas las operaciones de reenvío, (ii) anuncio de capacidades, (iii) informes de logística, y (iv) notificación de eventos. El valor de SDN se basa en la expectativa de que el CDPI se implementa de forma abierta, neutral e interoperable por el proveedor.

2.1.5. Interfaces norte SDN (Northbound Interfaces NBI)

Las NBI SDN son interfaces entre aplicaciones SDN y controladores SDN y, por lo general, proporcionan vistas de red abstractas y permitir la expresión directa del comportamiento y los requisitos de la red. Esto puede ocurrir en cualquier nivel de abstracción (latitud) y en diferentes conjuntos de longitudes de funcionalidad). Uno de los valores de SDN reside en la expectativa de que estas interfaces se implementen de manera abierta, neutral e interoperable por el proveedor.

2.1.6. Controladores de interfaz y agentes

Cada interfaz se implementa mediante un par de controlador-agente, el agente representa el lado "sur", el lado inferior o la cara orientada a la infraestructura y el conductor el lado "norte", la parte superior o el lado de la aplicación

2.1.7. Administración

El plano de administración cubre las tareas estáticas que son mejores manejados fuera de la aplicación, control y planos de datos. Los ejemplos incluyen la gestión de relaciones comerciales entre el proveedor y el cliente, la asignación de recursos a los clientes, configuración de equipos físicos, coordinación de accesibilidad y credenciales entre entidades lógicas y físicas, configuración de arranque. Cada entidad empresarial tiene sus propias entidades de gestión. La comunicación entre entidades de gestión está más allá del alcance de esta arquitectura SDN. Uno de los objetivos de SDN es subsumir muchas tareas de administración conocidas de la red heredada en el CDPI. La Figura 2.2 muestra en forma resumida la arquitectura general de SDN

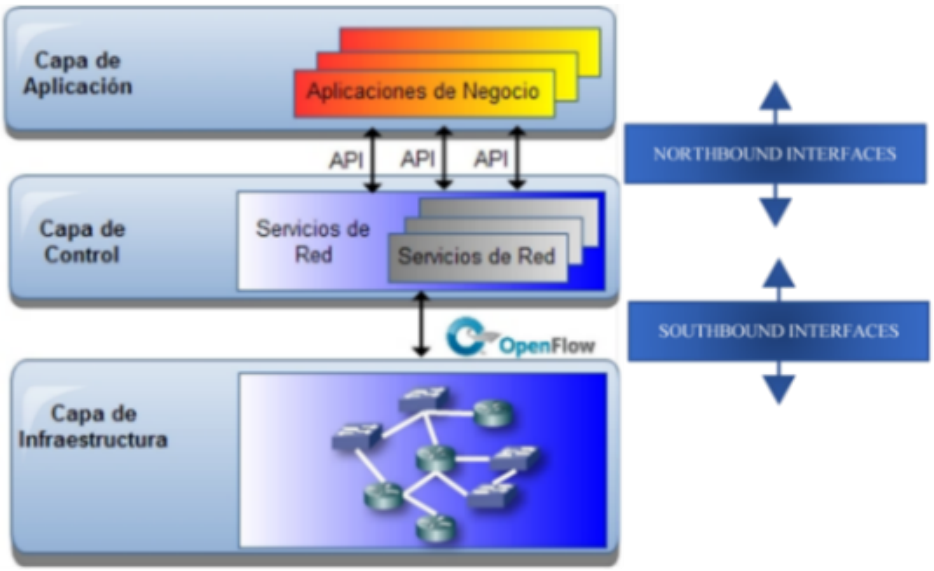

Figura 2.2: Arquitectura SDN resumida -Adaptado de [8]

2.2. Descripción de los planos operacionales la arquitectura SDN

La Figura 2.3, describe la arquitectura SDN desde el punto de vista de:

- a) Planos operacionales
- b) Componentes de cada plano operacional
- c) Separación o fronteras entre planos

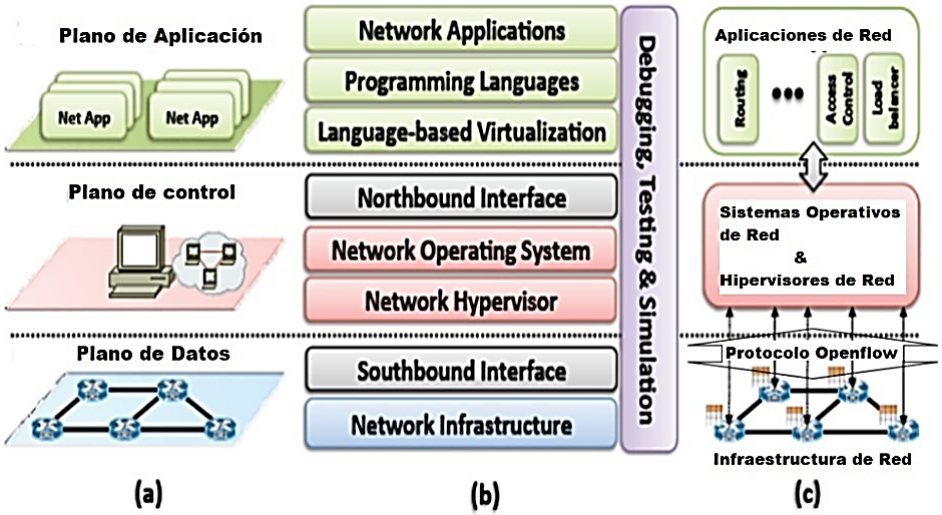

Figura 2.3: Arquitectura SDN clasificada funcionalmente

2.2.1. Componentes De Los Planos Operacionales

2.2.1.1. Northbound Interfaces

Una interfaz hacia el norte sigue siendo una cuestión abierta, es muy pronto para estandarizarla, se basará en la experiencia con los diferentes controladores. Además, se debe permitir que las aplicaciones de red no dependan de implementaciones específicas. Los sistemas de gestión externa o las aplicaciones de red (Net Apps) pueden desear extraer información sobre la red subyacente o controlar un aspecto del comportamiento o la política de la red. Además, los controladores pueden encontrar que es necesario comunicarse entre sí por una variedad de razones. Por ejemplo, una aplicación de control interno puede necesitar reservar recursos a través de múltiples dominios de control, o un controlador primario puede necesitar compartir información de políticas con un controlador de respaldo. A diferencia de la comunicación controlador-switch (que es la interfaz hacia el sur), actualmente no existe un estándar aceptado para la interfaz hacia el norte y es más probable que se implementen de manera ad-hoc para aplicaciones particulares. Una posible razón es que la interfaz hacia el norte se define completamente en el software, mientras que las interacciones controlador-switch deben habilitar la implementación del hardware. Si consideramos el controlador como un sistema operativo de red, entonces debería haber una interfaz claramente definida por la cual las aplicaciones puedan acceder al hardware subyacente (switches), coexistir e interactuar con otras aplicaciones, y utilizar los servicios del sistema (por ejemplo, descubrimiento de topología, reenvío, y así sucesivamente), sin requerir que el desarrollador de la aplicación conozca los detalles de implementación del controlador (que es el sistema operativo de red). Si bien existen varios controladores, sus interfaces de aplicación aún se encuentran en las primeras etapas e independientes entre sí e incompatibles. Hasta que surja un claro estándar de interfaz hacia el norte, las aplicaciones SDN continuarán desarrollándose de manera ad hoc y el concepto de aplicaciones de red flexibles y portátiles puede tener que esperar algún tiempo.

2.2.1.2. Sistema Operativo de Red / Controlador

SDN se comprometió a simplificar la gestión por medio del control de lógica centralizada que ofrece un sistema operativo de red o Controlador. De esta manera los desarrolladores ya no tienen que preocuparse de los detalles de bajo nivel en la distribución de datos entre los elementos de enrutamiento o la complejidad de desarrollar nuevos protocolos.

2.2.1.3. Hipervisores de Red

La virtualización ya es un hecho. Según informes recientes hay más servidores virtuales que servidores físicos. Los Hipervisores permiten a las máquinas virtuales compartir recursos de hardware. Esto es muy útil porque permite nuevos modelos de negocio en los que los usuarios asignan recursos bajo demanda de una infraestructura física compartida y los proveedores gestionan mejor su capacidad. Además, las máquinas virtuales permiten una fácil migración.

2.2.1.4. Southbound Interfaces

Son los puentes de conexión entre los elementos de control y los de reenvío, siendo el elemento esencial para separar la funcionalidad del plano de datos y de control. Las API están todavía muy enlazadas a dispositivos físicos de reenvío de paquetes en una infraestructura física. Un switch puede

tardar bastante tiempo de desarrollo para estar presente en el mercado, OpenFlow es uno de los primeros estándares de interfaz de comunicación definida entre las capas de control y forwarding en una arquitectura SDN. Además, permite el acceso directo y la manipulación de la capa de forwarding de los dispositivos de Red (switches, router), ya sean físicos o virtuales (basadas en un Hypervisor).

2.2.1.5. Estándar de Referencia Protocolo OpenFlow

Aunque en los siguientes puntos se describe con todo detalle, la idea base de OpenFlow es simple, se explota el hecho de que la mayor parte de los switches y routers Ethernet contienen tablas de flujo (típicamente construidas de TCAMs), que corren en línea para implementar firewalls, NATs, QoS (Quality of Service) y para recolectar estadísticas, pese a que la tabla de cada vendedor es diferente, se identificaron un conjunto interesante de funciones que corren en muchos switches y routers, el objetivo principal de OpenFlow es explotar estas funcionalidades comunes. OpenFlow es un protocolo abierto para programar las tablas de flujo en diferentes switches y routers, un administrador de red puede dividir el tráfico en flujos de producción e investigación. Los investigadores controlan sus propios flujos al escoger las rutas que sus paquetes siguen y el procesamiento que reciben; en este sentido los investigadores pueden probar y experimentar con nuevos protocolos, nuevos modelos de seguridad, esquemas de direccionamiento e incluso alternativas a la tradicional IP; mientras en la misma red el tráfico de producción es aislado y procesado de forma tradicional y estándar.

2.2.1.6. Capa Infraestructura

De manera similar a las redes tradicionales, las SDN se componen de un conjunto de equipos de red. La diferencia está en que ahora los dispositivos físicos son simples elementos de reenvío con software para tomar decisiones.

2.3. Arquitectura SDN para Redes de Transporte

Esta arquitectura describe un escenario que representa la interconexión de dominios SDN como una abstracción de estructura jerárquica de control (Controlador Padre-Hijo) ver Figura 2.4, la Open Networking Foundation en su documento SDN Architecture for Transport Networks [9] describe la arquitectura SDN para redes de transporte expresando: “Las redes de transporte deben adaptarse a la creciente demanda de ancho de banda de los centros de datos, respaldar el despliegue rápido de servicios por parte de los proveedores de servicios y brindar capacidad de respuesta en tiempo real a los cambios de capacidad / QoS. Los proveedores de aplicaciones y servicios desean la capacidad de solicitar y proporcionar conexiones de extremo a extremo con SLA garantizado (en términos de, por ejemplo, ancho de banda, demora, disponibilidad, rendimiento de errores) en múltiples tipos de infraestructuras de transporte, incluidas OTN, MPLS-TP y Carrier Ethernet. La arquitectura SDN y el modelo de información aplicado a las redes de transporte admiten características clave para el transporte, como la abstracción / virtualización y la relación de flujo y conexión". No entramos en detalles sobre esta arquitectura pues es como una “recursión"de los aspectos funcionales del plano de control que desarrollamos en nuestra investigación.

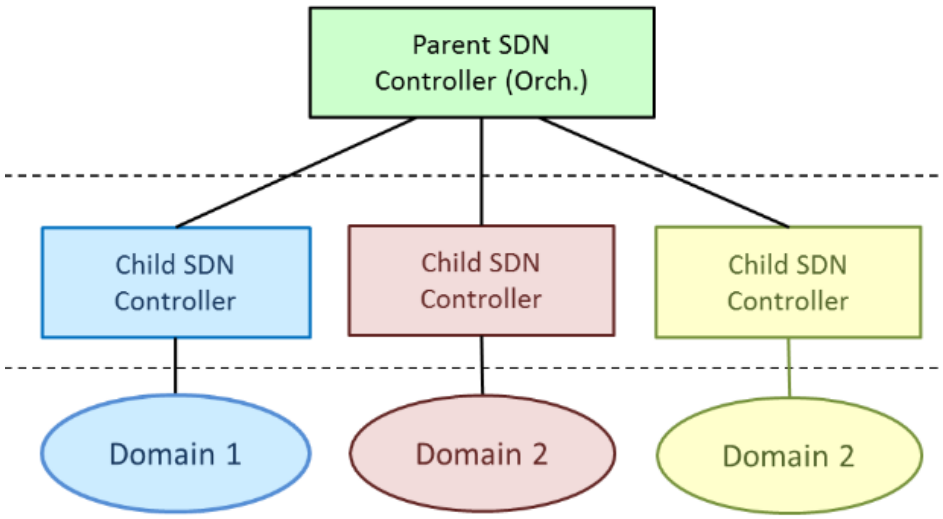

Figura 2.4: Controladores jerárquicos en una red multi-dominio SDN -Tomado de [9]

2.3.1. Eastbound / Westbound Interfaces

Es interesante resaltar como que no existe un estándar para las API en Northbound (orientado a la aplicación), y del mismo modo tampoco se estandarizaron aún protocolos de las APIs este / oeste. (Eastbound / Westbound). En cuanto a las API este / oeste, es interesante la lectura del artículo de William Stallings. “Software-Defined Networks and OpenFlow"[10], donde expresa “En una red empresarial grande, la implementación de un solo controlador para administrar todos los dispositivos de red resultaría difícil de manejar o indeseable. Un escenario más probable es que el operador de una gran empresa o red de transporte divida toda la red en numerosos dominios SDN no superpuestos, como se muestra en la Figura 2.5. Un ejemplo interesante es SDNi [62] se trata de un protocolo en

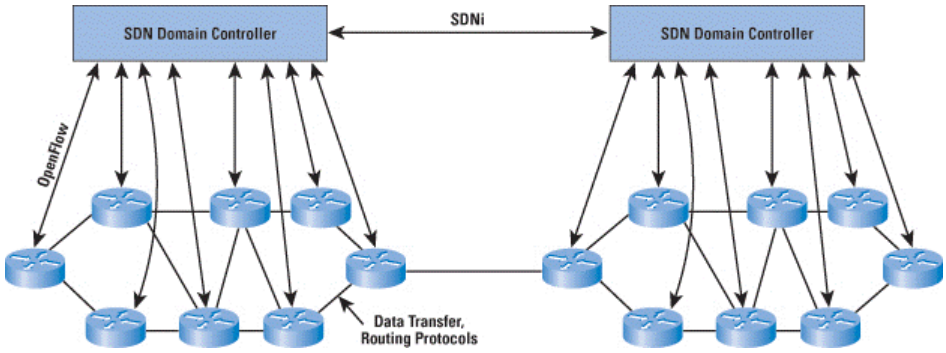

Figura 2.5: Planos Operacionales de control y Datos en una arquitectura SDN para Red de Transporte [10]

desarrollo de la Internet Research Task Force, para la interconexión de dominios SDN. Es responsable de coordinar comportamientos entre controladores SDN e intercambio de información de control y aplicación a través de múltiples dominios SDN. SDNi es un protocolo de “EastBound Interface / WestBound Interface.[en]tre los controladores SDN, haciendo una analogía con OpenFlow que es un protocolo de la Southbound Interface.

2.4. El Protocolo OPENFLOW – Especificaciones

La Open Networking Foundation publica todas las especificaciones técnicas que se van elaborando y actualizando con relación al protocolo OpenFlow, las mismas están disponibles en línea a través de su sitio web “Technical Specifications of the Open Networking Foundation"[46], Las especificaciones técnicas incluyen todos los estándares que definen un protocolo, modelo de información, funcionalidad de componentes y documentos marco relacionados. La versión que aquí se describe es “Version 1.3.1 (Wire Protocol 0x04)"[7] , la versión 1.0 es muy utilizada por razones de compatibilidad, pero a partir de la versión 1.3 se agregaron funcionalidades como manejo de múltiples tablas y variedad de mejoras que pueden verse al final de la especificación [7]. A la fecha en que esto se escribe se encuentran publicadas hasta la “Version 1.5.1 (Protocol version 0x06)"[11] de marzo de 2015, que incluye muchas actualizaciones funcionales y administrativas. No obstante, es necesario destacar que las implementaciones de los fabricantes y desarrolladores, en su gran mayoría, todavía no aplican más allá de la versión 1.3, incluso como ya se mencionó todavía se utiliza 1.0

2.4.1. Switch de Referencia OpenFlow

El switch OpenFlow es un dispositivo de reenvío básico, al que se puede acceder mediante el protocolo OpenFlow. La OpenFlow Switch Specification Version 1.3.1 [7] cubre los componentes y las funciones básicas del switch, y el protocolo OpenFlow para administrar un switch OpenFlow desde un controlador remoto. Un Switch OpenFlow consta de una o más tablas de flujo y una tabla de grupo, que realiza búsquedas de paquetes y reenvío, y un canal OpenFlow a un controlador externo Figura 2.6. El switch se comunica con el controlador y el controlador administra el switch a través del protocolo OpenFlow.

2.4.2. Componentes Switch OpenFlow

Usando el protocolo OpenFlow, el controlador puede agregar, actualizar y eliminar entradas de flujo en las tablas de flujo, tanto de manera reactiva (en respuesta a los paquetes) como de manera proactiva. Cada tabla de flujo en el switch contiene un conjunto de entradas de flujo; cada entrada de flujo consiste en campos de coincidencia (“matching"), contadores y un conjunto de instrucciones para aplicar a paquetes coincidentes (ver [7] sección 5.2). La coincidencia comienza en la primera tabla de flujo y puede continuar con tablas de flujo adicionales (ver [7] sección 5.1). Las entradas de flujo comparan los paquetes en orden de prioridad con la primera entrada coincidente en cada tabla que se está utilizando (ver [7] sección 5.3). Si se encuentra una entrada coincidente, se ejecutan las instrucciones asociadas con la entrada de flujo específica. Si no se encuentra una coincidencia en una tabla de flujo, el resultado depende de la configuración de la entrada de Table-Miss (ver [7] sección 5.1), por ejemplo:

- El paquete puede reenviarse al controlador a través del canal OpenFlow
- Descartarse
- Puede continuar a la siguiente tabla de flujo (ver [7] 5.4).

Las instrucciones asociadas con cada entrada de flujo contienen acciones o modifican el procesamiento de la CANALIZACIÓN (ver [7] 5.9). Las acciones incluidas en las instrucciones describen:

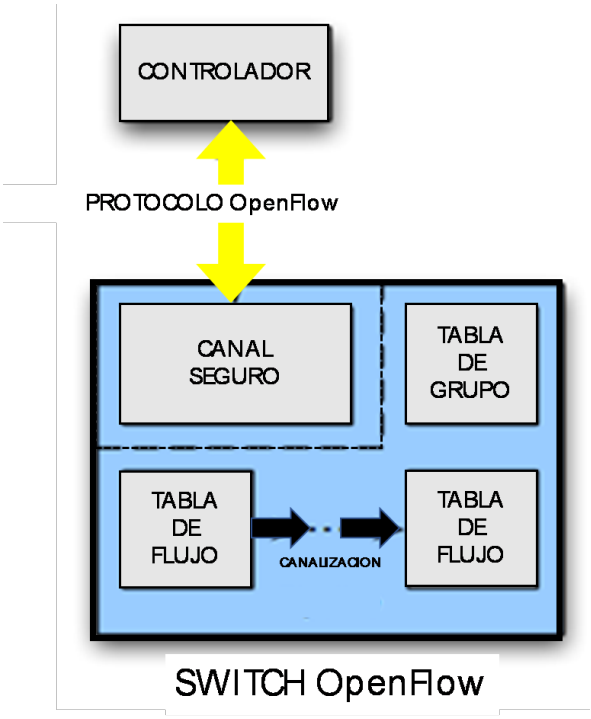

Figura 2.6: Componentes de un switch OpenFlow - Adaptado de [7]

- El reenvío de paquetes
- La modificación de paquetes
- Tabla de grupos.

Las instrucciones de procesamiento de canalización permiten que los paquetes se envíen a tablas posteriores para su procesamiento, y permiten que la información, en forma de metadatos, se comunique entre las tablas. El procesamiento de canalización se detiene cuando el conjunto de instrucciones asociado con una entrada de flujo coincidente no especifica una siguiente tabla. En este punto, el paquete generalmente se modifica y reenvía (ver [7] 5.10).

Las entradas de flujo pueden reenviar a un puerto. Generalmente, este es un puerto físico, pero también puede ser un puerto lógico definido por el switch o un puerto reservado definido por esta especificación (consulte [7] 4.1). Los puertos reservados pueden especificar acciones genéricas de reenvío, como el envío al controlador, la inundación (flooding), o el reenvío utilizando métodos que no son de OpenFlow, como el procesamiento "normal"del switch (ver [7] 4.5), mientras que los puertos lógicos definidos por el switch pueden especificar grupos de agregación de enlaces, túneles o interfaces loopback (ver [7] 4.4).

Las acciones asociadas con las entradas de flujo también pueden dirigir los paquetes a un grupo que especifica un procesamiento adicional (ver [7] 5.6). Los grupos representan conjuntos de acciones para inundación (flooding), así como una semántica de reenvío más compleja (por ejemplo, multi-ruta, redireccionamiento rápido y agregación de enlaces). Como capa general de direccionamiento indirecto, los grupos también permiten que múltiples entradas de flujo se reenvíen a un único identificador (por ejemplo, reenvío de IP a un salto siguiente común). Esta abstracción permite que las acciones de salida comunes en las entradas de flujo se cambien de manera eficiente.

La tabla de grupo contiene entradas de grupo; cada entrada de grupo contiene una lista de conjuntos de acciones (action buckets), con semántica específica que dependen del tipo de grupo (ver [7] 5.6.1). Las acciones en uno o más action buckets, se aplican a los paquetes enviados al grupo. Los diseñadores de switches tienen la libertad de implementar los elementos internos en cualquier forma conveniente, siempre que se conserven la semántica correcta de instrucción y coincidencia. Por ejemplo, mientras que una entrada de flujo puede usar un grupo “all"para reenviar a múltiples puertos, un diseñador de switches puede elegir implementar esto como una sola máscara de bits dentro de la tabla de reenvío de hardware. Otro ejemplo es “matching"; la canalización expuesta por un switch OpenFlow puede implementarse físicamente con un número diferente de tablas de hardware.

2.4.3. Puertos OpenFlow

Los puertos OpenFlow son las interfaces de red para pasar paquetes entre el procesamiento de OpenFlow y el resto de la red. Los switch OpenFlow se conectan lógicamente entre sí a través de sus puertos OpenFlow. Un switch OpenFlow hace que una serie de puertos OpenFlow estén disponibles para el procesamiento de OpenFlow. El conjunto de puertos OpenFlow puede no ser idéntico al conjunto de interfaces de red provisto por el hardware del switch, algunas interfaces de red pueden estar deshabilitadas para OpenFlow, y el switch OpenFlow puede definir puertos OpenFlow adicionales. Los paquetes de OpenFlow se reciben en un puerto de ingreso y son procesados por la canalización de OpenFlow (ver [7] sección 5.1) que puede reenviarlos a un puerto de salida. El puerto de ingreso de paquetes es una propiedad del paquete a lo largo de la canalización de OpenFlow y representa el puerto de OpenFlow en el que se recibió el paquete en el switch OpenFlow. El puerto de ingreso se puede usar cuando se emparejan paquetes (ver [7] sección 5.3). La canalización de OpenFlow puede decidir enviar el paquete en un puerto de salida utilizando la acción de salida (ver [7] sección 5.12), que define cómo el paquete vuelve a la red. Un switch OpenFlow debe admitir tres tipos de puertos OpenFlow: Puertos físicos puertos lógicos y puertos reservados.

2.4.4. Puertos Estándar

Los puertos estándar de OpenFlow se definen como puertos físicos, puertos lógicos y el puerto reservado LOCAL si se admite (excluyendo otros puertos reservados). Los puertos estándar se pueden usar como puertos de entrada y salida, se pueden usar en grupos (ver [7] sección 5.6) y tienen contadores de puertos (ver [7] sección 5.8).

2.4.5. Puertos Físicos

Los puertos físicos de OpenFlow son puertos definidos por el switch que corresponden a una interfaz de hardware del switch. Por ejemplo, en un switch Ethernet, los puertos físicos se asignan uno a uno a las interfaces Ethernet. En algunas implementaciones, el switch OpenFlow puede virtualizarse sobre el hardware del switch. En esos casos, un puerto físico OpenFlow puede representar un segmento virtual de la interfaz de hardware correspondiente del switch

2.4.6. Puertos Lógicos

Los puertos lógicos de OpenFlow son puertos definidos por el switch que no corresponden directamente a una interfaz de hardware del switch. Los puertos lógicos son abstracciones de nivel superior

que pueden definirse en el switch, utilizando métodos que no son OpenFlow (por ejemplo, grupos de agregación de enlaces, túneles, interfaces de bucle invertido). Los puertos lógicos pueden incluir encapsulación de paquetes, y se pueden asignar a varios puertos físicos. El procesamiento realizado por el puerto lógico debe ser transparente para el procesamiento de OpenFlow y esos puertos deben interactuar con el procesamiento de OpenFlow como los puertos físicos de OpenFlow. Las únicas diferencias entre puertos físicos y los puertos lógicos es que un paquete asociado con un puerto lógico puede tener un campo de metadatos adicional llamado ID de túnel asociado con él (ver [7] subsección A.2.3.7) y cuando un paquete recibido en un puerto lógico se envía al controlador, ambos el puerto lógico y su puerto físico subyacente se informan al controlador (consulte [7] subsección A.4.1).

2.4.7. Puertos Reservados

Los puertos reservados de OpenFlow están definidos por la especificación OpenFlow. Especifican acciones de reenvío genéricas, como el envío al controlador, la inundación (flooding), o el reenvío (Forwarding), utilizando métodos que no son de OpenFlow, como el procesamiento "normal"del switch. No se requiere que un switch tenga que admitir todos los puertos reservados, solo aquellos marcados como Requerido.[a] continuación.

2.4.7.1. ALL (Requerido)

Representa todos los puertos que el switch puede usar para reenviar un paquete específico. Solo se puede utilizar como puerto de salida. En ese caso, se envía una copia del paquete a todos los puertos estándar, excluyendo el puerto de ingreso de paquetes y los puertos que están configurados como OFPPC_NO_FWD.

2.4.7.2. CONTROLLER (Requerido)

Representa el canal de control con el controlador OpenFlow. Puede utilizarse como puerto de entrada o como puerto de salida. Cuando se utiliza como puerto de salida, encapsula el paquete en un mensaje Packet-In y lo envía al controlador, usando el protocolo OpenFlow (vea [7] A.4.). Cuando se utiliza como puerto de entrada, identifica un paquete que se origina en el controlador y es enviado al switch Packet-out.

2.4.7.3. TABLE (Requerido)

Representa el inicio de la canalización de OpenFlow. Este puerto solo es válido en una acción de salida en la lista de acciones de un mensaje de salida de paquete, y envía el paquete a la primera tabla de flujo para que el paquete pueda procesarse a través de la canalización regular de OpenFlow.

2.4.7.4. IN PORT (Requerido)

Representa el puerto de entrada de paquetes. Solo se puede utilizar como un puerto de salida, envía el paquete mediante su puerto de entrada.

2.4.7.5. ANY (Requerido)

Valor especial utilizado en algunos comandos de OpenFlow cuando no se especifica ningún puerto (puerto con comodín). No se puede utilizar como puerto de entrada ni como puerto de salida.

2.4.7.6. LOCAL (Opcional)

Representa la pila de red local del switch y su pila de administración. Puede utilizarse como puerto de entrada o como puerto de salida. El puerto local permite que las entidades remotas interactúen con el switch y sus servicios de red a través de la red OpenFlow, en lugar de hacerlo a través de una red de control independiente. Con un conjunto adecuado de entradas de flujo predeterminadas, se puede utilizar para implementar una conexión de controlador en banda.

2.4.7.7. NORMAL (Opcional)

Representa la canalización tradicional no OpenFlow del switch. Solo se puede usar como un puerto de salida y procesa el paquete utilizando la canalización normal. Si el switch no puede reenviar paquetes de la canalización de OpenFlow a la canalización normal, debe indicar que no admite esta acción.

2.4.7.8. FLOOD (Opcional)

Representa la inundación utilizando la canalización normal del switch (consulte [7] 5.1). Solo se puede usar como un puerto de salida, en general enviará el paquete a todos los puertos estándar, pero no al puerto de ingreso, o los puertos que están en el estado OFPPS_BLOCKED. El switch también puede usar el paquete VLAN ID para seleccionar qué puertos inundar. Los switches OpenFlow solo, no admiten el puerto NORMAL y Puerto FLOOD, mientras que los switches OpenFlow-híbridos pueden admitirlos (ver [7] 5.1). el reenvío de paquetes al puerto FLOOD depende de la implementación y configuración del switch, mientras se reenvía utilizando un grupo de tipo ALL, permite que el controlador implemente de manera más flexible la inundación (ver [7] 5.6.1).

2.4.8. Funcionamiento del Protocolo OpenFlow en dispositivos compatibles

Antes de entrar en la descripción detallada de la especificación OpenFlow, es conveniente tener una visión resumida del funcionamiento del protocolo y el contexto en que lo hace. Históricamente, para dar solución a los requisitos que surgen habitualmente en las arquitecturas de las redes en Data Center; los switches convencionales para resolver e implementar la necesaria redundancia de enlaces de estas redes, tradicionalmente acudieron a (por citar algunos) protocolos como Spanning-Tree (STP) [27] o algunas de sus versiones mejoradas (RSTP) [63], Shortest Path Bridging (SPB) [29], o Transparent Interconnection of Lots of Links (TRILL) [30], para determinar cómo se reenvían los paquetes. En los dispositivos OpenFlow-hibrido, es decir aquellos que aceptan la operación ethernet normal y también operación OpenFlow, el protocolo OpenFlow se utiliza para trasladar esa decisión de reenvío desde los switches a los controladores, el controlador normalmente reside en una computadora o estación de trabajo. Una aplicación de gestión se ejecutará en las interfaces del controlador que une todos los switches en la red, facilitando la configuración de caminos de reenvío que utilizarán todo el ancho de banda disponible. La especificación OpenFlow 1.3.1 [7], define el protocolo entre el controlador y los

switches y un conjunto de operaciones que se pueden realizar entre ellos Las instrucciones de reenvío se basan en el flujo, que consiste en que todos los paquetes comparten una serie de características comunes. Obviamente esto también funciona de la misma forma en dispositivos OpenFlow-solo, nada más que en éstos existe solamente la operación OpenFlow.

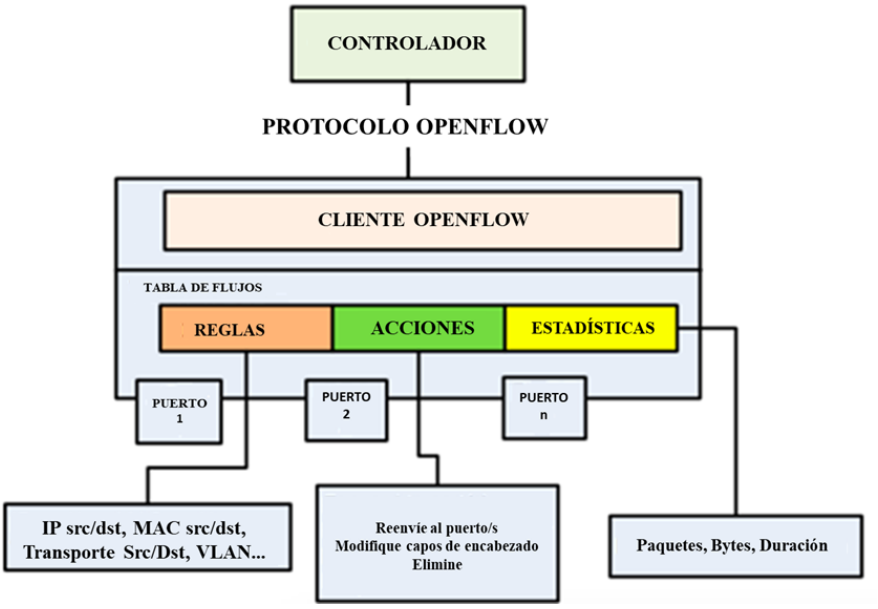

Figura 2.7

Cuando un paquete que llega al switch, si no encuentra ninguna coincidencia en ninguna de las entradas de la tabla correspondiente Un se debe crear un nuevo flujo (ver Figura 2.7). El switch debería tener configurado un descarte (Drop) de paquetes para un flujo que no haya sido definido con una entrada, pero sucede que, en la mayoría de los casos, el paquete será enviado al controlador mediante un mensaje Packet-in. El controlador entonces define un nuevo flujo para ese paquete y crea una o más entradas para la tabla. A continuación, el controlador envía la entrada o entradas al switch para que sean añadidas a sus tablas de flujo mediante un mensaje Packet-out. el cual contiene el paquete original, el mismo buffer_id que contenía el Packet-in y una acción que indica lo que hará con el paquete. Cada flujo de entrada tiene asociada una acción simple, las tres básicas son:

- Reenvío de flujo de paquetes a un puerto o puertos dados. Esto permite que los paquetes ser encaminados a través de la red. En la mayoría de los switches sucede a la velocidad de la línea.
- Encapsulación y reenvío este flujo de paquetes al controlador. El paquete será enviado al Canal Seguro, donde se encapsula y se envía al controlador. Típicamente se usa para el primer paquete en un nuevo flujo, para que el controlador pueda decidir si el flujo debe ser añadido a la tabla de flujos. También se puede usar para reenviar todos los paquetes al controlador para que sean procesados.
- Descartar este flujo de paquetes. Puede ser usado por seguridad, para parar ataques de denegación de servicio o reducir el falso tráfico de descubrimiento broadcast desde los hosts finales.

Existen infinidad de parámetros que pueden especificarse para definir un flujo. Entre los posibles criterios podemos incluir los puertos por donde se reciben los paquetes cuando llegan, el puerto Ethernet

de origen, la etiqueta VLAN, el destino Ethernet o el puerto IP y otro número de características de los paquetes. A continuación (punto 2.5) desarrollo de la descripción que hace la especificación OpenFlow.

2.5. Tablas del Protocolo OpenFlow

Esta sección describe los componentes de las tablas de flujo y las tablas de grupo, junto con la mecánica de "matching"(coincidencia de los paquetes con entradas en la tabla de flujo) y las acciones aplicadas a los paquetes.

2.5.1. Proceso de Canalización (Pipeline)

Los dispositivos compatibles con OpenFlow, también llamados switches OpenFlow pueden ser de dos tipos: OpenFlow-solo: Estos dispositivos solo admiten la operación OpenFlow, en esos dispositivos todos los paquetes son procesados por la canalización de OpenFlow, y no pueden procesarse de otra manera. OpenFlow-híbrido: Estos dispositivos admiten tanto la operación OpenFlow como la operación Ethernet normal. Es decir, la conmutación Ethernet L2 tradicional, el aislamiento de VLAN, el enrutamiento L3 (enrutamiento IPv4, enrutamiento IPv6 ...), el procesamiento de ACL y QoS. Esos dispositivos deberían proporcionar un mecanismo de clasificación fuera de OpenFlow que enrute el tráfico a la canalización de OpenFlow o la canalización normal. Por ejemplo, un switch puede usar la etiqueta VLAN o el puerto de entrada del paquete para decidir si procesa el paquete usando una canalización o la otra, o puede dirigir todos los paquetes a la canalización OpenFlow. Este mecanismo de clasificación está fuera del alcance de la especificación [7]. Un switch OpenFlow-híbrido también puede permitir que un paquete pase de la canalización de OpenFlow a la canalización normal a través de los puertos reservados NORMAL y FLOOD (ver [7] sección 4.5). La canalización OpenFlow de cada dispositivo OpenFlow contiene varias tablas de flujo, cada tabla de flujo contiene varias entradas de flujo. El procesamiento de canalización de OpenFlow define cómo interactúan los paquetes con esas tablas de flujo (consulte la Figura 2.8). Se requiere que un switch OpenFlow tenga al menos una tabla de flujo, y opcionalmente puede tener más tablas de flujo. Un switch OpenFlow con una sola tabla de flujo es válido, en este caso el procesamiento de la canalización se simplifica enormemente.

Para ilustrar mejor esto y entender mejor el proceso se muestra en la Figura 2.9 lo que menciona la norma OpenFlow 1.5.1 respecto del proceso de canalización y está más completa que en la versión 1.3.1.

“la canalización de cada switch lógico OpenFlow contiene una o más tablas de flujo, cada tabla de flujo contiene múltiples entradas de flujo. El procesamiento de la canalización OpenFlow define cómo los paquetes interactúan con esas tablas de flujo (Figura 2.10). Se requiere que un switch de OpenFlow tenga al menos una tabla de flujo de ingreso, y opcionalmente puede tener más tablas de flujo. Un switch OpenFlow con una sola tabla de flujo es válido, en este caso el procesamiento de la canalización se simplifica enormemente. Las tablas de flujo de un switch OpenFlow están numeradas en el orden en que pueden ser atravesadas por paquetes, partiendo de 0. El procesamiento de canalización se realiza en dos etapas, el procesamiento de ingreso y el procesamiento de egreso. La separación de las dos etapas se indica mediante la primera tabla de egreso (ver [11] 7.3.2), todas las tablas con un número menor que la primera tabla de egreso deben usarse como tablas de ingreso, y ninguna tabla con un número mayor o igual que la primera tabla de egreso se puede usar como una tabla de ingreso. El procesamiento de la canalización siempre comienza con el procesamiento de ingreso en la primera tabla de flujo: el

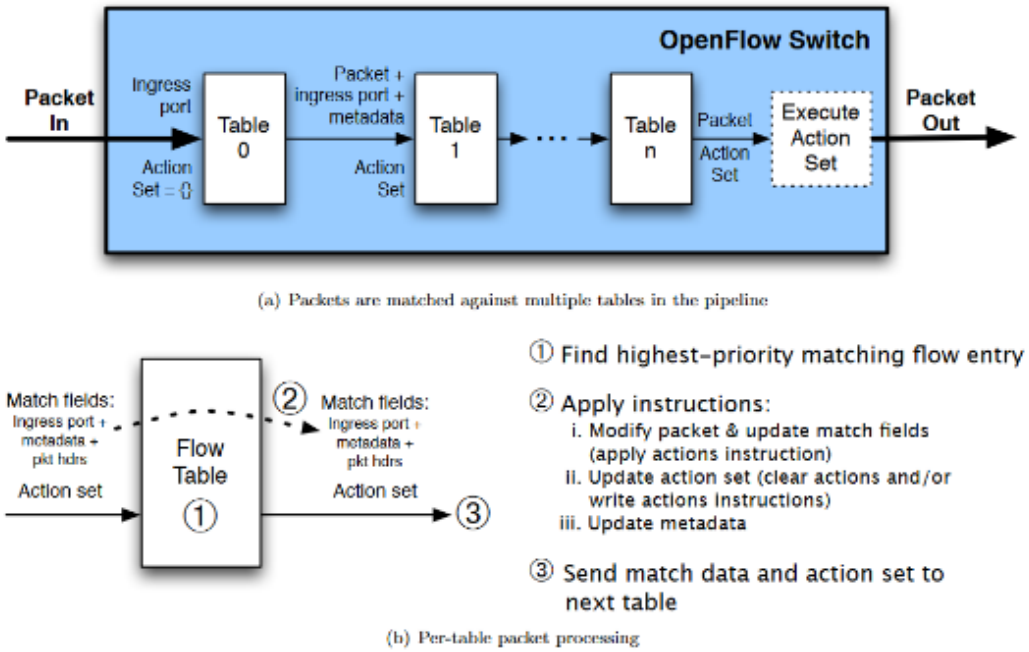

Figura 2.8: Procesamiento de Canalización (Pipeline) de la versión 1.3 [7]

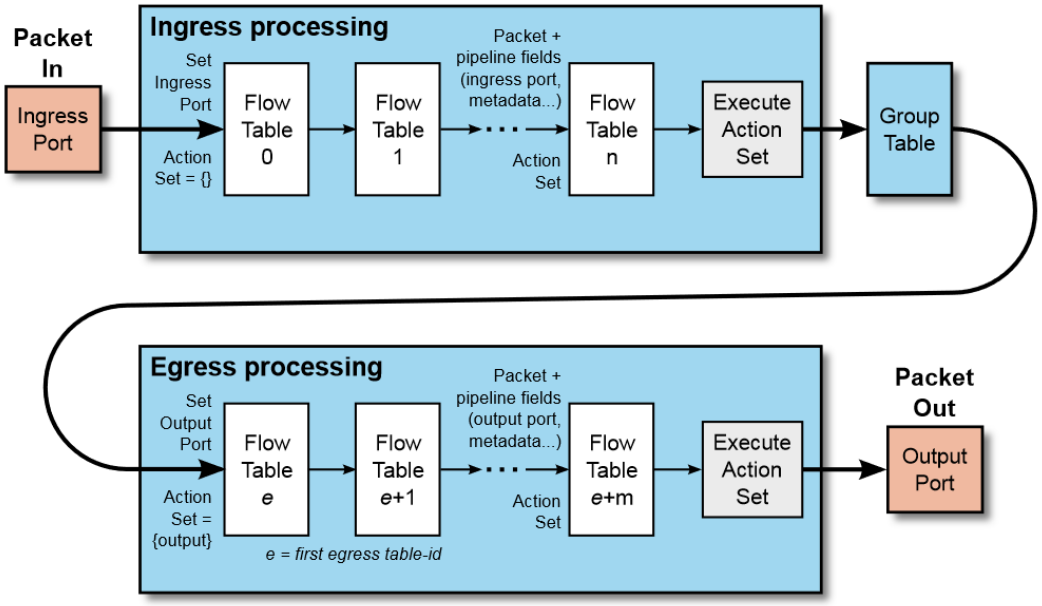

Figura 2.9: Secuencia de eventos de los paquetes a través de la canalización (de OpenFlow V1.5.1)

paquete debe coincidir primero con las entradas de flujo de la tabla de flujo 0 (consulte la Figura 1.3). Se pueden usar otras tablas de flujo de ingreso dependiendo del resultado de la coincidencia en la primera tabla. Si el resultado del procesamiento de ingreso es reenviar el paquete a un puerto de salida, el switch OpenFlow puede realizar el procesamiento de egreso en el contexto de ese puerto de salida. El procesamiento de egreso es opcional, un switch puede no admitir ninguna tabla de egreso o puede no estar configurado para usarlos. Si no se configura una tabla de egreso válida como la primera tabla de egreso (consulte [11] 7.3.2), el puerto de salida debe procesar el paquete y, en la mayoría de los casos, el paquete se reenvía fuera del switch. Si una tabla de egreso válida se configura como la primera tabla de egreso (ver [11] 7.3.2), el paquete debe compararse con las entradas de flujo de esa tabla, y se pueden usar otras tablas de egreso dependiendo del resultado del matching (coincidencia) en esa tabla".

2.5.2. Tablas de flujos

La Figura 2.11, muestra la tabla de flujos correspondiente al protocolo OpenFlow versión 1.3.1

- Match Fields: Coincidencia. Este campo consiste en el puerto y la cabecera de paquete y opcionalmente metadatos especificados en una tabla anterior.
- Priority: Prioridad del flujo. En el caso de que haya más de un flujo con el mismo match tendrá prioridad el flujo con el número de prioridad más alto.
- Counters: Contadores que se actualizan cuando los paquetes coinciden (matched).
- Instructions: Instrucciones que indican la acción que hará el paquete a procesar.
- Timeouts: tiempos antes de que un flujo expire. Son dos: idle_time_out y hard_time_out. El primero hacer referencia al tiempo que un flujo estará sin usar. El segundo indica el tiempo de vida real, partiendo desde que se instaló el flujo.
- Cookie: Valor que elige el controlador para filtrar las estadísticas, las modificaciones y el borrado de los flujos. No se usa: cuando se procesan los paquetes.

2.5.3. Proceso de Ingreso de Un Paquete (Matching)

Según la versión OpenFlow 1.3.1, al recibir un paquete, un switch OpenFlow realiza las funciones que se muestran en la Figura 2.12, El switch comienza realizando una búsqueda en la primera tabla de flujo y, según el procesamiento de la canalización, puede realizar búsquedas en otras tablas de flujo.

-Los campos de coincidencia de paquetes se extraen del paquete. Los campos de coincidencia utilizados para las búsquedas de tablas dependen del tipo de paquete y, por lo general, incluyen varios campos de encabezado, como la dirección de origen de Ethernet o la dirección de destino de IPv4 (ver [7] subsección A.2.3). Además de los campos en encabezados de paquetes, las coincidencias también se pueden realizar en el puerto de ingreso y en los campos de metadatos. Los metadatos se pueden usar para pasar información entre tablas en un switch. Los campos de coincidencia de paquete representan el paquete en su estado actual, si existen acciones se aplicaron en una anterior tabla. -La tabla que utiliza Apply-Actions cambia los encabezados de los paquetes, esos cambios se reflejan en los campos de coincidencia de paquetes. Un paquete coincide con una entrada de la tabla de flujo si los valores en los campos de coincidencia utilizados para la búsqueda coinciden con los definidos en la entrada de la tabla de flujo. Si un campo de entrada de la tabla de flujo tiene un valor de ANY (campo omitido), coincide con todos los valores posibles en el encabezado. Si el switch admite máscaras de bits arbitrarias en

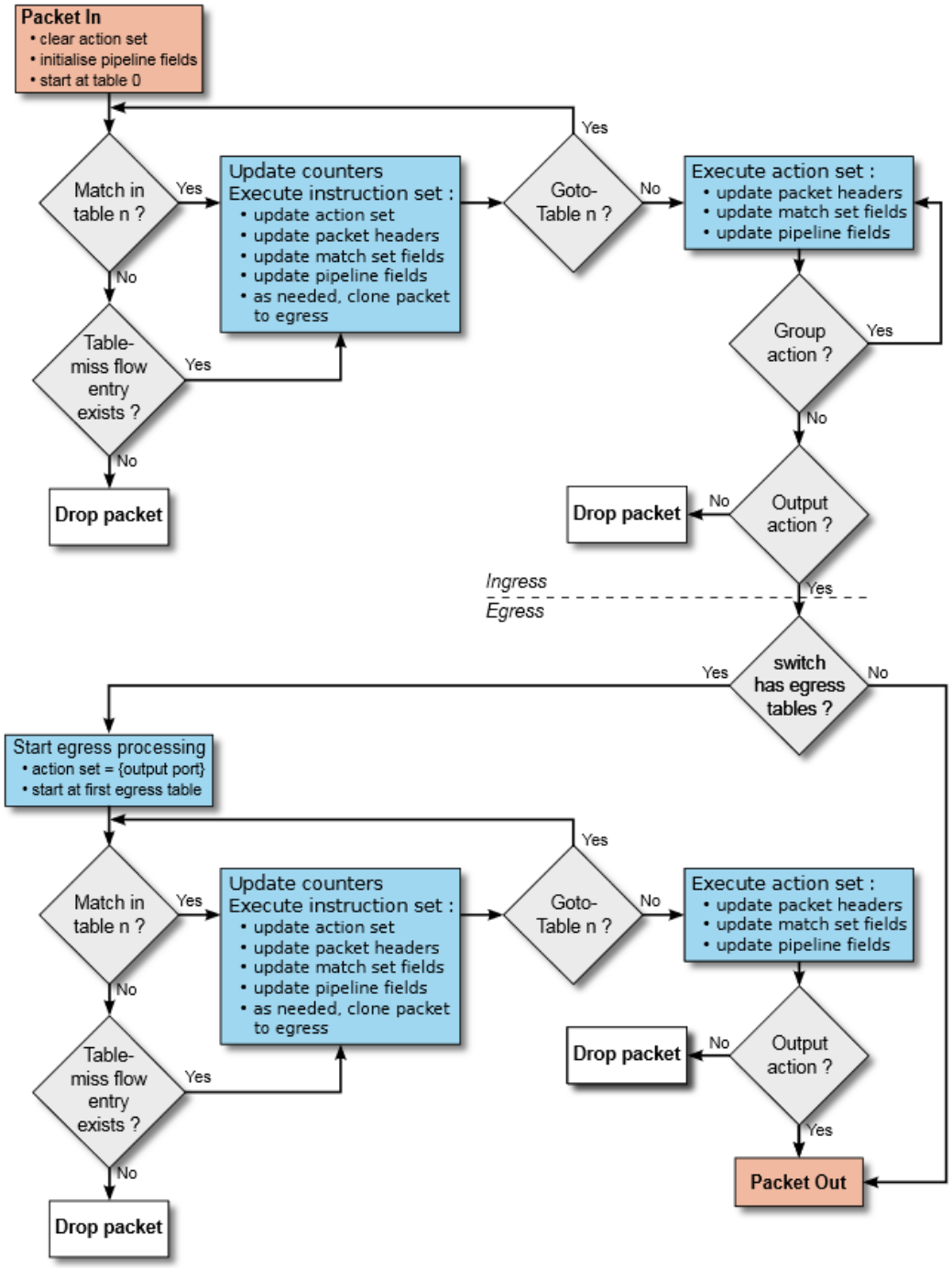

Figura 2.10: Diagrama de flujo simplificado que detalla el flujo de paquetes a través de un switch OpenFlow [11]

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie

Figura 2.11: Flujos OpenFlow versión 1.3.1

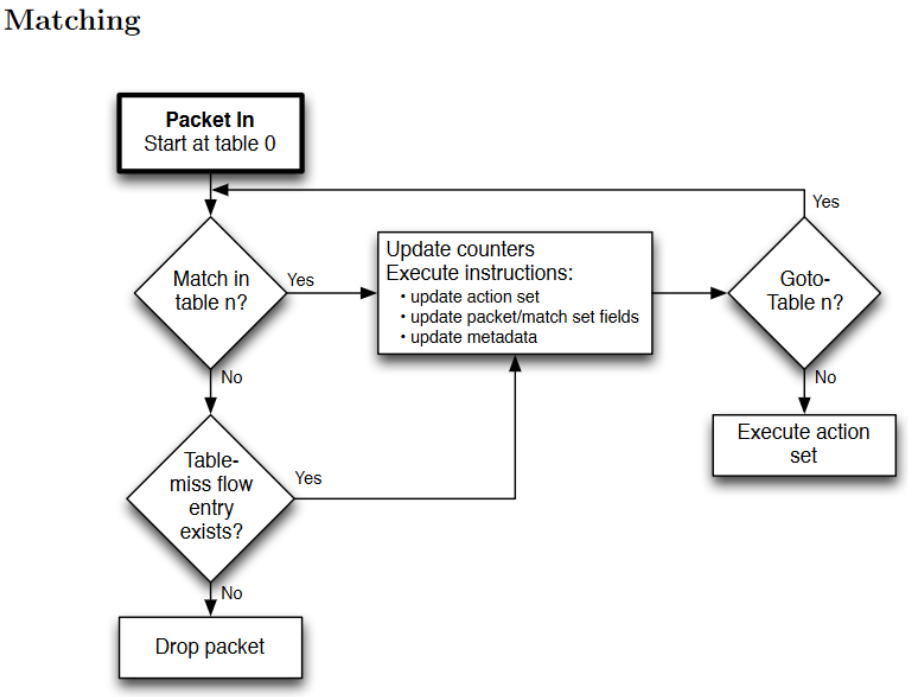

Figura 2.12: Proceso de Matching [7]

campos de coincidencia específicos, estas máscaras pueden especificar coincidencias con mayor precisión. El paquete se compara con la tabla y solo se debe seleccionar la entrada de flujo de mayor prioridad que coincida con el paquete. Los contadores asociados con la entrada de flujo seleccionada deben actualizarse y el conjunto de instrucciones incluido en la entrada de flujo seleccionada debe aplicarse. Si hay múltiples entradas de flujo coincidentes con la misma prioridad más alta, la entrada de flujo seleccionada está explícitamente indefinida. Este caso solo puede surgir cuando un escritor controlador nunca establece el bit OFPFF_CHECK_OVERLAP en mensajes de modificación de flujo y agrega entradas superpuestas. -Los fragmentos de IP se deben volver a ensamblar antes del procesamiento de la canalización si la configuración del switch contiene el indicador OFPC_FRAG_REASM (consulte [7] A.3.2). -Esta versión [7] de la especificación, no define el comportamiento esperado cuando un switch recibe un paquete con formato incorrecto o dañado

2.5.4. Table-Miss

Cada tabla de flujo debe admitir una entrada Table-Miss para procesar las fallas de tabla. La entrada Table-Miss especifica cómo procesar paquetes no coincidentes con las otras entradas de flujo de la tabla (ver [7] sección 5.1), y puede, por ejemplo, enviar paquetes al controlador, eliminar paquetes o dirigir paquetes a la siguiente tabla. La entrada Table-Miss, se identifica por su coincidencia y su prioridad (ver [7] sección 5.2), tiene carácter comodín en todos los campos (se omiten todos los campos) y tiene la prioridad más baja (0). La coincidencia de entrada de Table-Miss puede quedar fuera del rango normal de coincidencias admitido por una tabla de flujo, por ejemplo, una tabla de coincidencia exacta no admite comodines para otras entradas de flujo, pero debe admitir el comodín de entrada Table-Miss en todos los campos. Es posible que la entrada de Table-Miss no tenga la misma capacidad que una entrada de flujo regular (consulte [7] subsección A.3.5.5). Se recomienda que las

implementaciones admitan entradas de flujo con errores, con la misma capacidad que el procesamiento de errores de tabla de versiones anteriores de OpenFlow, es decir:

- envíe paquetes al controlador
- descarte paquetes
- dirija paquetes a una siguiente tabla.

La entrada Table-Miss se comporta en la mayoría de los casos, del mismo modo que cualquier otra entrada de la tabla: no existe de forma predeterminada en una tabla, el controlador puede agregarla o eliminarla en cualquier momento (consulte [7] 6.4) y puede caducar (ver [7] 5.5). La entrada Table-Miss coincide con los paquetes de la tabla como se espera de su conjunto de campos de coincidencia y prioridad (ver [7] 5.3), coincide con paquetes no coincidentes con otras entradas en la tabla. Las instrucciones de entrada Table-Miss se aplican a los paquetes que coinciden con la entrada Table-Miss (ver [7] 5.9). Si la entrada de Table-Miss envía directamente paquetes al controlador utilizando el puerto CONTROLLER (ver [7] 4.5), del Packet-in debe identificar la razón como una entrada Table-Miss (ver [7] subsección A.4.1). Si la entrada de Table-Miss no existe, de forma predeterminada, los paquetes que no coinciden con ninguna de las entradas de flujo en una tabla son descartados (Drop), Una configuración de switch, por ejemplo, utilizando el Protocolo de configuración de OpenFlow, puede anular este valor predeterminado y especificar otro comportamiento. La Figura 2.10, muestra el diagrama de funcionamiento desde la entrada de un paquete en un switch OpenFlow.

2.5.5. Eliminación de flujo (Flow Removal)

Las entradas de flujo se eliminan de las tablas de flujo de dos maneras, ya sea a petición del controlador o mediante el mecanismo de expiración del flujo del switch. El mecanismo de expiración del flujo del switch que se ejecuta por el switch independientemente del controlador y se basa en el estado y la configuración de las entradas de flujo. Cada entrada de flujo tiene un idle_timeout y un hard_timeout asociado. Si cualquiera de los valores es distinto de cero, el switch debe tener en cuenta el tiempo de llegada de la entrada de flujo, ya que puede necesitar desalojar la entrada más tarde. Un campo hard_timeout distinto de cero hace que la entrada de flujo se elimine después de un número de segundos dado, independientemente de cuántos paquetes haya coincidido. Un campo idle_timeout distinto de cero hace que la entrada de flujo se elimine cuando no haya coincidido con ningún paquete en un dado número de segundos. El switch debe implementar la caducidad del flujo y eliminar las entradas de flujo de la tabla de flujo cuando se exceda uno de sus tiempos de espera. El controlador puede eliminar activamente las entradas de flujo de las tablas de flujo enviando mensajes de modificación de la tabla de flujo de eliminación (OFPFC_DELETE o OFPFC_DELETE_STRICT-ver [7] 6.4). Cuando se elimina una entrada de flujo, ya sea por el controlador o el mecanismo de expiración de flujo, el switch debe verificar el indicador OFPFF_SEND_FLOW_REM de la entrada de flujo. Si se establece este indicador, el switch debe enviar un mensaje de flujo eliminado al controlador. Cada mensaje de flujo eliminado contiene una descripción completa de la entrada de flujo, el motivo de la eliminación (caducidad o eliminación), la duración de la entrada de flujo en el momento de la eliminación y las estadísticas de flujo en el momento de la eliminación.

2.5.6. Tablas de Grupos

Los grupos proveen una eficiente manera de indicar que el mismo conjunto de acciones deben ser llevadas a cabo por múltiples flujos. Por ello, es útil para implementar multicast y unicast.

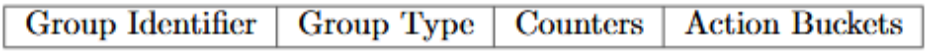

Figura 2.13: Principales componentes de una entrada de Tabla de Grupos [7]

Cada entrada consiste en su identificador, el tipo de grupo, un contador y un set de action buckets. (Figura 2.13).

- Identificador de grupo: entero sin signo que define unívocamente el grupo.
- Tipo de Grupo: para determinar la semántica del grupo.
- Contadores: Actualizados cuando los paquetes son procesados por el grupo.
- Set de acciones (action buckets): lista ordenada de acciones, donde cada acción contiene un conjunto de acciones a ejecutar y unos parámetros asociados.

2.5.7. Tablas Meter

La Tabla Meter, consiste en un conjunto de entradas con medidas definiendo medidas por flujo.

Con estas tablas, se mide la tasa de paquetes asignadas a ellas y se facilita el control de la tasa de esos paquetes. Están unidas directamente a las entradas de flujo (al contrario que las colas que están relacionadas con los puertos). Consiste en entradas de medidas, definiendo medidas por flujo. Viabilizan que OpenFlow pueda implementar operaciones de QoS simples, como medición basada en DSCP que puede clasificar un conjunto de paquetes en múltiples categorías según su tasa. Los medidores son totalmente independientes de las colas por puerto (ver 5.8), sin embargo, en muchos casos, estas dos características se pueden combinar para implementar trabajos complejos conservando frameworks de QoS, como Diff Serv. Una tabla meter, tiene los siguientes campos (ver Figura 2.14):

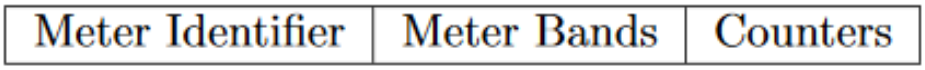

Figura 2.14: Campos de tabla meter, tomado de [7]

- Meter Identifier: entero sin signo de 32 bits que define unívocamente el medidor.
- Meter Bands: lista ordenada de bandas, donde cada una especifica la tasa y la manera de procesar el paquete.
- Counters: campo que se actualiza cuando el paquete es procesado por alguna de ellas.

Para ilustrar y entender mejor se incluye lo siguiente tomado de la especificación OpenFlow V1.5.1 "Las diferentes entradas de flujo en la misma tabla pueden usar: el mismo medidor, diferentes medidores o ningún medidor. Al usar diferentes medidores en una tabla, conjuntos desunidos de entradas de flujo se puede medir de forma independiente. Los paquetes pueden pasar por varios medidores cuando se usan medidores en tablas de flujo sucesivas, en cada tabla de flujo la entrada de flujo correspondiente puede dirigirlo a una medida. Otra forma de usar varios medidores, si el switch lo admite, es usar múltiples acciones de medidor por entrada. Esto se puede usar para realizar una medición jerárquica, en la que

los vértices para controlar los flujos se miden de forma independiente y en conjunto (consulte Figura 2.15)".

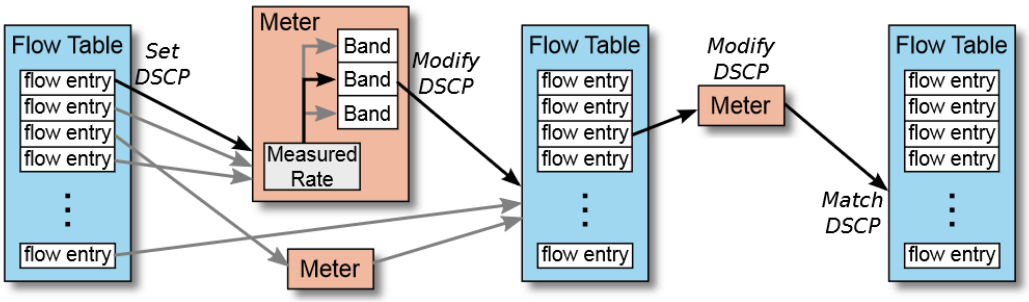

Figura 2.15: Medidores y medición jerárquica DSCP [11]

2.5.7.1. Meters Band

Cada medidor puede tener una o más bandas de medida. Cada banda especifica la tasa que aplica y la forma en que deben procesarse los paquetes. Los paquetes se procesan en una sola banda en función de su tasa actual, el medidor se aplica la banda de medidor con la tasa más alta configurada y que es inferior a la tasa de medición actual. Si la tasa actual es más baja que cualquier tasa de banda de medidor especificada, no se aplica ninguna banda de medidor. Cada banda se identifica por su tasa y en la Figura 2.16 y la Figura 2.17 se muestran las diferencias entre versiones de OpenFlow y es un buen ejemplo de cómo se va mejorando la funcionalidad del protocolo.

Band Type	Rate	Counters	Type specific arguments

Figura 2.16: Meters Band, de la especificación V1.3.1 [7]

Band Type	Rate	Burst	Counters	Type specific arguments

Figura 2.17: Meters Band, de la especificación V1.5.1 [11]

- Band Type (tipo de banda): define cómo se procesan los paquetes
- Rate (tasa): utilizada por el medidor para seleccionar la banda del medidor, define la tasa más baja a la que se puede aplicar la banda
- Counters (contadores): actualizados cuando los paquetes son procesados por una banda de medidor
- Burst (ráfaga): define la granularidad de la banda del medidor
- Type specific arguments (tipo de argumentos específicos): algunos tipos de banda tienen argumentos opcionales

No hay un tipo de banda Requerido"por esta especificación. El controlador puede consultar al switch sobre cuál de los tipos de Meter Band “Opcional.es compatible

- Opcional: Drop (descartar el paquete). Puede utilizarse para definir una banda limitadora de velocidad.

- Opcional: remarcado de dscp: disminuye la prioridad de descarte del campo DSCP en el encabezado IP del paquete. Se puede usar para definir una política simple en Diff Serv.

2.5.8. Counters

Los contadores se mantienen para cada tabla de flujo, entrada de flujo, puerto, cola, grupo, grupo de grupos, medidor y banda de medidor. Los contadores compatibles con OpenFlow pueden implementarse en software y mantenerse contando los contadores de hardware con rangos más limitados. La Figura 2.18, contiene el conjunto de contadores definidos por la especificación Open-Flow 1.3.1. Un dispositivo o switch, no requiere admitir todos los contadores, solamente aquellos marcados como Requeridos.[en] la Figura 2.18. La duración se refiere a la cantidad de tiempo que se ha instalado la entrada de flujo, un puerto, un grupo, una cola o un medidor en el switch, y debe medirse con una precisión de segundos. El campo Recibir errores es el total de todos los errores de recepción y colisión definidos en la figura 2-8, así como cualquier otro que no se indique en la tabla. Los contadores están sin firmar y se envuelven (wrap around) sin indicador de desbordamiento. Si un contador numérico específico no está disponible en el switch, su valor debe establecerse en el valor máximo del campo (el equivalente sin signo de -1).

Counter	Bits	
Per Flow Table		
Reference count (active entries)	32	*Required*
Packet Lookups	64	*Optional*
Packet Matches	64	*Optional*
Per Flow Entry		
Received Packets	64	*Optional*
Received Bytes	64	*Optional*
Duration (seconds)	32	*Required*
Duration (nanoseconds)	32	*Optional*
Per Port		
Received Packets	64	*Required*
Transmitted Packets	64	*Required*
Received Bytes	64	*Optional*
Transmitted Bytes	64	*Optional*
Receive Drops	64	*Optional*
Transmit Drops	64	*Optional*
Receive Errors	64	*Optional*
Transmit Errors	64	*Optional*
Receive Frame Alignment Errors	64	*Optional*
Receive Overrun Errors	64	*Optional*
Receive CRC Errors	64	*Optional*
Collisions	64	*Optional*
Duration (seconds)	32	*Required*
Duration (nanoseconds)	32	*Optional*
Per Queue		
Transmit Packets	64	*Required*
Transmit Bytes	64	*Optional*
Transmit Overrun Errors	64	*Optional*
Duration (seconds)	32	*Required*
Duration (nanoseconds)	32	*Optional*
Per Group		
Reference Count (flow entries)	32	*Optional*
Packet Count	64	*Optional*
Byte Count	64	*Optional*
Duration (seconds)	32	*Required*
Duration (nanoseconds)	32	*Optional*
Per Group Bucket		
Packet Count	64	*Optional*
Byte Count	64	*Optional*
Per Meter		
Flow Count	32	*Optional*
Input Packet Count	64	*Optional*
Input Byte Count	64	*Optional*
Duration (seconds)	32	*Required*
Duration (nanoseconds)	32	*Optional*
Per Meter Band		
In Band Packet Count	64	*Optional*
In Band Byte Count	64	*Optional*

Figura 2.18: Lista de Contadores (Counters) [7]

2.5.9. Instructions

Cada entrada de flujo contiene un conjunto de instrucciones que se ejecutan cuando un paquete coincide con la entrada. Estas instrucciones dan como resultado cambios en el paquete, el conjunto de acciones y / o el procesamiento de la canalización. No se requiere que el switch tenga que admitir todos los tipos de instrucción, solamente aquellas marcadas como Ïnstrucción requerida.[a] continuación. El controlador también puede consultar el switch acerca de cuál de las ïnstrucciones opcionales.[es] compatible.

- Meter meter id (opcional): paquete directo al medidor especificado. Como resultado de la medición, el paquete puede caerse.
- Apply-Actions action(s) (opcional): aplica las acciones específicas de inmediato, sin ningún cambio en el Conjunto de acciones (Action Set). Esta instrucción se puede usar para modificar el paquete entre dos tablas o para ejecutar varias acciones del mismo tipo. Las acciones se especifican como una lista de acciones (Action list) (ver [7] sección 5.11).
- Clear-Actions (opcional): borra todas las acciones en el conjunto de acciones inmediatamente.
- Write-Actions action(s) (requerida): fusiona las acciones especificadas en el conjunto de acciones actual (ver [7] sección 5.10). Si existe una acción del tipo dado en el conjunto actual, sobrescríbala, de lo contrario, agréguela.
- Write-Metadata metadata / mask (opcional): escribe el valor de metadatos enmascarados en el campo de metadatos. La máscara especifica qué bits del registro de metadatos deben modificarse (es decir, nuevos metadatos = metadatos antiguos y máscara | valor y máscara)
- Goto-Table next-table-id (requerida): indica la siguiente tabla en el proceso de procesamiento. La tabla-id debe ser mayor que la tabla-id actual. Las entradas de flujo de la última tabla de la canalización no pueden incluir esta instrucción (ver [7] sección 5.1).

El conjunto de instrucciones asociado con una entrada de flujo contiene un máximo de una instrucción de cada tipo. Las instrucciones del conjunto se ejecutan en el orden especificado en esta lista anterior. En la práctica, las únicas restricciones son que las instrucciones Meter, se ejecuten antes de las instrucciones Apply-Actions, las instrucciones Clear-Actions, se ejecuten antes que las instrucciones Write-Actions, y por último se ejecuta Goto-Table. Un switch debe rechazar una entrada de flujo si no puede ejecutar las instrucciones asociadas con la entrada de flujo. En este caso, el switch debe devolver un error de flujo no soportado (consulte [7] 6.4). Las tablas de flujo pueden no ser compatibles con cada coincidencia, cada instrucción o cada acción.

2.5.10. Action-Set

Un conjunto de acciones está asociado con cada paquete. Este conjunto está vacío por defecto. Una entrada de flujo puede modificar el conjunto de acciones utilizando una instrucción Write-Action o una instrucción Clear-Action, asociada con un matching (coincidencia) en particular. El conjunto de acciones se realiza entre tablas de flujo. Cuando el conjunto de instrucciones de una entrada de flujo no contiene una instrucción Goto-Table, el procesamiento de la canalización se detiene y se ejecutan las acciones en el conjunto de acciones del paquete. Una instrucción Set asociada a una entrada de flujo, contiene máximo una instrucción de cada tipo. Las acciones Set-Field se identifican por sus tipos de campo, por lo tanto, el Action-Set contiene un máximo de una acción de Set-Field para cada tipo de campo (es decir, se pueden configurar múltiples campos). Cuando se requieren múltiples acciones del mismo tipo, por ejemplo, presionando múltiples etiquetas MPLS o abriendo múltiples etiquetas MPLS, se puede usar la instrucción Aplicar Acciones (Apply-Actions) (ver [7] 5.11). Las acciones en un Action-Set se aplican en el orden especificado a continuación, independientemente del orden en que se agregaron al conjunto. Si un conjunto de acciones contiene un grupo de acciones (group action), las acciones en el apropiado grupo de acciones, también se aplican en el orden especificado a continuación. El switch puede soportar ejecución en orden arbitrario mediante una lista de acciones Action-List de la instrucción Apply-Actions.

1º) copy TTL inwards: aplica las acciones internas de copia TTL al paquete.

2º) pop: aplica todas las acciones pop a la etiqueta del paquete.
3º) push-MPLS: aplica la acción push a la etiqueta del paquete MPLS.
4º) push-PBB: aplica la acción push a la etiqueta PBB del paquete.
5º) push-VLAN: aplica la acción push a la etiqueta VLAN del paquete.
6º) copy TTL outwards: aplica acción copiar TTL al paquete hacia el exterior.
7º) decrement TTL: aplica la acción decrementar TTL al paquete
8º) set: aplica todas las acciones set-field al paquete
9º) qos: aplica todas las acciones de QoS, como ser: set queue al paquete
10º) group: Si se especifica una acción de grupo, aplica las acciones del grupo o grupos relevantes en el orden especificado en esta lista.
11º) output: si no se especifica ninguna acción de grupo, reenvíe el paquete en el puerto especificado por la acción de salida

La acción de salida en el conjunto de acciones se ejecuta en último lugar. Si tanto una acción de salida como una acción de grupo se especifican en un conjunto de acciones, la acción de salida se ignora y la acción de grupo tiene prioridad. Si no se especificaron acciones de salida ni acciones de grupo en un conjunto de acciones, el paquete se descarta. La ejecución de grupos es recursiva si el switch lo admite; un grupo de grupos puede especificar otro grupo, en cuyo caso la ejecución de acciones atraviesa todos los grupos especificados por la configuración del grupo.

2.5.11. Actions-List

La instrucción Apply-Actions y el mensaje de salida del paquete (Packet-out), incluyen una lista de acciones. La semántica de la lista de acciones es idéntica a la especificación OpenFlow 1.0 [54] Las acciones de una lista de acciones se ejecutan en el orden especificado por la lista y se aplican inmediatamente al paquete. La ejecución de una lista de acciones comienza con la primera acción en la lista y cada acción se ejecuta en el paquete en secuencia. El efecto de esas acciones es acumulativo, si la lista de acciones contiene dos acciones de Push VLAN, se agregan dos encabezados de VLAN al paquete. Si la lista de acciones contiene una acción de salida, una copia del paquete se reenvía en su estado actual al puerto deseado. Si la lista contiene acciones de grupo, una copia del paquete es procesada en su estado actual por los grupos relevantes. Después de la ejecución de la lista de acciones en una instrucción Apply-Action, la ejecución de la canalización continúa en el paquete modificado (ver [7] sección 5.1). El conjunto de acciones del paquete no se modifica por la ejecución de la acción.

2.5.12. Actions

No se requiere que el switch tenga que admitir todos los tipos de acciones mencionadas a continuación, solo los marcados como requerida". El controlador también puede consultar al switch acerca de con cuál de las .ªcciones opcionales.ᵉˢ compatible.

- Output (requerida): La acción output envía un paquete a un puerto OpenFlow especificado (ver [7] 4.1). Los switches OpenFlow deben admitir el reenvío a puertos físicos, puertos lógicos definidos por el switch y los puertos reservados requeridos (consulte [7] 4.5).
- Set-Queue (opcional): La acción set-queue establece el ID de la cola para un paquete. Cuando el paquete se reenvía a un puerto utilizando la acción de salida, el identificador de la cola determina qué cola adjunta a este puerto se usa para programar y reenviar el paquete. El comportamiento

de reenvío es dictado por la configuración de la cola y se usa para proporcionar soporte básico de Calidad de Servicio (QoS) (ver [7] subsección A.2.2).

- Drop (requerida): No hay una acción explícita para representar Drop(descarte). En su lugar, los paquetes cuyos conjuntos de acciones no tienen acciones de salida deben eliminarse. Este resultado podría provenir de conjuntos de instrucciones vacíos o grupos de acciones vacíos en el procesamiento, o después de ejecutar una instrucción Clear-Actions.
- Group (requerida): Procesa el paquete a través del grupo especificado. La interpretación exacta depende del tipo de grupo.
- Push-Tag / Pop-Tag (opcional): Los switches pueden admitir la capacidad de utilizar etiquetas push / pop como se muestra en la Figura 2.19.

Para facilitar la integración con las redes existentes, se sugiere que se admita la capacidad de etiquetado push / pop VLAN. Las etiquetas más nuevas siempre deben insertarse como la etiqueta más externa en la ubicación de las etiquetas. Cuando se inserta una nueva etiqueta VLAN, debe ser la etiqueta más externa insertada, inmediatamente después del encabezado de Ethernet y antes de otras etiquetas; del mismo modo que se hace en MPLS, cuando se inserta una nueva etiqueta, debería ser la Etiqueta más externa insertada, inmediatamente después del encabezado de Ethernet y antes de otras etiquetas. Cuando se agregan múltiples etiquetas push al conjunto de acciones del paquete, las acciones se aplican al paquete en el orden definido por las reglas del conjunto de acciones, primero, MPLS, luego PBB, luego VLAN (ver [7] 5.10). Cuando se incluyen varias acciones de inserción en una lista de acciones, se aplican al paquete en el orden de la lista (ver [7] 5.11). **Nota**: consulte [7] la

Action	Associated Data	Description
Push VLAN header	Ethertype	Push a new VLAN header onto the packet. The Ethertype is used as the Ethertype for the tag. Only Ethertype 0x8100 and 0x88a8 should be used.
Pop VLAN header	-	Pop the outer-most VLAN header from the packet.
Push MPLS header	Ethertype	Push a new MPLS shim header onto the packet. The Ethertype is used as the Ethertype for the tag. Only Ethertype 0x8847 and 0x8848 should be used.
Pop MPLS header	Ethertype	Pop the outer-most MPLS tag or shim header from the packet. The Ethertype is used as the Ethertype for the resulting packet (Ethertype for the MPLS payload).
Push PBB header	Ethertype	Push a new PBB service instance header (I-TAG TCI) onto the packet (see A.2.5). The Ethertype is used as the Ethertype for the tag. Only Ethertype 0x88E7 should be used.
Pop PBB header	-	Pop the outer-most PBB service instance header (I-TAG TCI) from the packet (see A.2.5).

Figura 2.19: Etiquetas Push/Pop actions [7]

sección 5.12.1 para obtener información sobre los valores de campo predeterminados.

- Set-Field (opcional). Las diversas acciones de Set-Field se identifican por su tipo de campo y modifican los valores de los respectivos campos de encabezado en el paquete. Si bien no es estrictamente necesario, la compatibilidad con la reescritura de varios campos de encabezado con acciones Set-Field aumenta en gran medida la utilidad de una implementación de OpenFlow. Para ayudar a la integración con las redes existentes, sugerimos que las acciones de modificación de VLAN sean soportadas. Las acciones de Set-Field siempre deben aplicarse al encabezado más externo posible (por ejemplo, una acción “Establecer ID de VLAN"siempre establece el ID de la etiqueta de VLAN más externa), a menos que el tipo de campo especifique lo contrario.

- Change-TTL (opcional). Las diversas acciones de Change-TTL modifican los valores del campo: TTL en IPv4, Hop Limit en IPv6 o TTL en MPLS. Si bien no es estrictamente necesario, las acciones que se muestran en la Figura 2.20, aumentan en gran medida la utilidad de una implementación de OpenFlow para implementar funciones de enrutamiento. Las acciones de cambio-TTL siempre deben aplicarse al encabezado más externo posible.

Action	Associated Data	Description
Set MPLS TTL	8 bits: New MPLS TTL	Replace the existing MPLS TTL. Only applies to packets with an existing MPLS shim header.
Decrement MPLS TTL	-	Decrement the MPLS TTL. Only applies to packets with an existing MPLS shim header.
Set IP TTL	8 bits: New IP TTL	Replace the existing IPv4 TTL or IPv6 Hop Limit and update the IP checksum. Only applies to IPv4 and IPv6 packets.
Decrement IP TTL	-	Decrement the IPv4 TTL or IPv6 Hop Limit field and update the IP checksum. Only applies to IPv4 and IPv6 packets.
Copy TTL outwards	-	Copy the TTL from next-to-outermost to outermost header with TTL. Copy can be IP-to-IP, MPLS-to-MPLS, or IP-to-MPLS.
Copy TTL inwards	-	Copy the TTL from outermost to next-to-outermost header with TTL. Copy can be IP-to-IP, MPLS-to-MPLS, or MPLS-to-IP.

Figura 2.20: Acciones Change-TTL [7]

El switch OpenFlow comprueba los paquetes con IP TTL o MPLS TTL no válidos y los rechaza. La comprobación de un TTL no válido no tiene que hacerse para cada paquete, pero se debe hacer como mínimo cada vez que se aplique una acción de disminución TTL a un paquete. La configuración asíncrona del switch puede modificarse (ver [7] 6.1.1) enviando paquetes con TTL no válido al controlador a través del canal de control mediante un mensaje Packet-in (ver [7] 6.1.2).

2.5.12.1. Valores por defecto para campos PUSH

Los valores de campo para todos los campos especificados en la Figura 2.21 deben copiarse de los encabezados externos existentes a los nuevos encabezados externos al ejecutar una acción de empuje. Los nuevos campos enumerados sin los campos existentes correspondientes deben establecerse en cero. Los campos que no se pueden modificar a través de las acciones del campo de juego de OpenFlow deben inicializarse con los valores de protocolo apropiados. Los campos en los nuevos encabezados pueden anularse especificando una acción de çonjunto"para el (los) campo (s) apropiado (s) después de la operación de inserción.

New Fields		Existing Field(s)
VLAN ID	←	VLAN ID
VLAN priority	←	VLAN priority
MPLS label	←	MPLS label
MPLS traffic class	←	MPLS traffic class
MPLS TTL	←	{ MPLS TTL / IP TTL
PBB I-SID	←	PBB I-SID
PBB I-PCP	←	VLAN PCP
PBB C-DA	←	ETH DST
PBB C-SA	←	ETH SRC

Figura 2.21: Campos existentes que pueden copiarse en nuevos campos en una acción PUSH [7]

2.6. OpenFlow Chanel

La comunicación entre el controlador y el switch se realiza mediante el protocolo OpenFlow, donde se puede intercambiar un conjunto de mensajes definidos entre estas entidades a través de un canal seguro. El canal seguro es la interfaz que conecta cada switch OpenFlow a un controlador. La conexión segura de la capa de transporte (TLS) al controlador definido por el usuario, se inicia cuando arranca el switch OpenFlow. El puerto TCP predeterminado del controlador es 6633. El switch y el controlador se autentican mutuamente intercambiando certificados firmados por una clave privada específica del sitio. Cada switch debe ser configurable por el usuario con un certificado para autenticar el controlador (certificado del controlador) y el otro para autenticar al controlador (certificado del switch). El canal OpenFlow es la interfaz que conecta cada switch OpenFlow a un controlador. A través de esta interfaz, el controlador configura y administra el switch, recibe eventos del switch y envía paquetes fuera del switch. Entre la ruta de datos y el canal OpenFlow, la interfaz es específica de la implementación, sin embargo, todos los mensajes del canal OpenFlow deben formatearse de acuerdo con el protocolo OpenFlow. El canal OpenFlow generalmente está cifrado mediante TLS, pero puede ejecutarse directamente sobre TCP.

2.7. Mensajes del protocolo OpenFlow

El protocolo contiene 3 tipos de mensajes:

- controlador a switch
- asíncrono
- simétrico

A continuación, un resumen de estos mensajes

2.7.1. Mensajes de Controlador a switch

Este tipo de mensajes son iniciados por el controlador y pueden o no requerir una respuesta desde el switch.

2.7.1.1. Features (Características)

Una vez establecida la sesión sobre el protocolo de seguridad de la capa de transporte (TLS;), el controlador envía un mensaje de petición de características (OFPT_FEATURES_REQUEST) al switch. El switch debe responder con una serie de características que respondan específicamente a las capacidades admitidas por este (OFPT_FEATURES_REPLY).

2.7.1.2. Configuration (Configuración)

El controlador es capaz de establecer y consultar los parámetros de configuración en el switch. El switch sólo responde a una pregunta formulada desde el controlador.

2.7.1.3. Modify-State (Modificar Estado)

Los mensajes Modify-State, son enviados por el controlador para gestionar el estado de los switches. Su propósito principal es añadir/eliminar y modificar flujos en las tablas de flujo y establecer las propiedades del puerto del switch. Estos mensajes son:

- o ADD: para las solicitudes ADD con el conjunto de indicadores OFPFF_CHECK_OVERLAP, el switch debe primero verificar si hay entradas de flujo superpuestas. Dos entradas de flujo se superponen si un solo paquete puede coincidir con ambas, y ambas entradas tienen la misma prioridad. Si existe un conflicto de superposición entre una entrada de flujo existente y la solicitud ADD, el switch debe rechazar la adición y responder con ofp_error_msg con el tipo de error FPET_FLOW_MODE_FAILED y el código de error OFPFMFC_OVERLAP. Para las solicitudes ADD no superpuestas válidas, o aquellas que no tienen una marca de verificación de superposición, el switch debe insertar la entrada de flujo en la tabla con el número más bajo para la cual el switch admite todos los comodines establecidos en la estructura flow_match, y para la cual se observaría la prioridad durante el proceso de coincidencia. Si una entrada de flujo con campos de encabezado y prioridad idénticos ya reside en la tabla de flujo, entonces esa entrada que incluye sus contadores debe eliminarse y la nueva entrada de flujo debe agregarse. Si un switch no puede encontrar ninguna entrada de la tabla para agregar la entrada de flujo entrante, el switch debe enviar ofp_error_msg con el tipo OFPET_FLOW_MOD_FAILD y el código de error PFOFMFC_ALL_TABLES_FULL. Si la lista de acciones en un mensaje de modificación de flujo hace referencia a un puerto no válido en un switch, el switch debe devolver ofp_error_msg con el tipo OFPET_BAD_ACTION y el código OFPBAC_BAD_OUT. Si el puerto al que se hace referencia puede ser válido en el futuro (por ejemplo, cuando se agrega una tarjeta de línea a un chasis), el switch puede eliminar los paquetes enviados al puerto de referencia o devolver inmediatamente un error OFPBAC_BAD_PORT y rechazar el mensaje de modificación de flujo.
- o MODIFY: si una entrada de flujo con un campo de encabezado idéntico no reside actualmente en la tabla de flujo, el comando MODIFY actúa como un comando ADD, y la nueva entrada de flujo debe insertarse con contadores ajustados a cero. De lo contrario, el campo de acciones se cambia en la entrada existente y sus contadores y los campos de tiempo de espera inactivo no se modifican.
- o DELETE: para solicitudes de eliminación, si no existe una entrada de flujo con identificador de grupo actual, entonces no se registra ningún error ni se produce ninguna modificación en

la tabla de flujo. De otra manera, el grupo es removido, y todas las entradas de flujo que contienen este grupo en un Action-group también se eliminan y luego cada entrada normal con el conjunto de indicadores OFPFF_SEND_FLOW_REM debe generar un mensaje de eliminación de flujo. Las entradas de flujo de emergencia eliminadas no generan mensajes de eliminación de flujo. Los comandos DELETE y DELETE_STRICT (ver siguiente punto) pueden ser filtrados opcionalmente por el puerto de salida. Si el campo out_port contiene un valor distinto de OFPP_NONE, introduce una restricción al hacer coincidir. Esta restricción es que la regla debe contener una acción de salida dirigida a ese puerto. Este campo es ignorado por los mensajes ADD, MODIFY y MODIFY STRICT. Para eliminar todos los grupos con un solo mensaje, especifique OFPG_ALL como valor del grupo.

- o MODIFY y DELETE: Estos comandos de modo de flujo tienen sus correspondientes versiones _STRICT. En las versiones no RESTRIC, los comodines están activos y todos los flujos que coinciden con la descripción se modifican o eliminan. En las versiones _STRICT, todos los campos, incluidos los comodines y la prioridad, que coinciden estrictamente con la entrada, solo se modifica o elimina un flujo idéntico. Por ejemplo, si se envía un mensaje para eliminar entradas al switch que tiene todas las banderas comodín configuradas, el comando DELETE eliminará todos los flujos de todas las tablas. Sin embargo, el comando DELETE_STRICT solo eliminaría una regla que se aplique a todos los paquetes con la prioridad especificada. Para el comando MODIFY y DELETE, NO STRICT, que contienen comodines, se producirá una coincidencia cuando una entrada de flujo coincida exactamente o sea más específica que la descripción en el comando flow_mod. Por ejemplo, si un comando DELETE dice que se eliminen todos los flujos con un puerto de destino 80, entonces no se eliminará una entrada de flujo que tenga todos los comodines. Sin embargo, si un comando DELETE tiene todos los comodines eliminará una entrada que coincida con todo el tráfico del puerto 80.

2.7.1.4. Read-State (Leer Estado)

Los mensajes Read-State son utilizados por el controlador para recopilar estadísticas de las tablas de flujo (flow-tables) del switch, de los puertos y entradas de flujo (flow-entries) individuales.

2.7.1.5. Packet-out

El controlador utiliza mensajes Packet-out para enviar paquetes desde un puerto específico en el switch y para reenviar los paquetes recibidos a través de los mensajes Packet-in. Los mensajes Packet-out deben contener un paquete completo o un ID de búfer que haga referencia a un paquete almacenado en el switch. El mensaje también debe contener una lista de acciones que se aplicarán en el orden en que se especifican, una lista de acciones vacía, o un Drop (descarte del paquete).

2.7.1.6. Barrier (Barrera)

Los mensajes Barrier, pueden ser de solicitud /respuesta (request/reply), y son utilizados por el controlador para garantizar que las dependencias de los mensajes se han cumplido o para recibir notificaciones de operaciones completadas.

2.7.1.7. Role-Request

El controlador utiliza los mensajes Role-Request para establecer la función de su canal OpenFlow o consultar esa función. Esto es útil sobre todo cuando el switch se conecta a varios controladores (ver [7] 6.3.4).

2.7.1.8. Asynchronous-Configuration

El controlador utiliza el mensaje Asynchronous-Configuration para establecer un filtro adicional en los mensajes asíncronos que desea recibir en su canal OpenFlow, o para consultar ese filtro. Esto es principalmente útil cuando el switch se conecta a múltiples controladores (ver [7] 6.3.4) y se realiza comúnmente al establecer el canal OpenFlow

2.7.2. Mensajes Asíncronos

Este tipo de mensajes son iniciados por el switch y utilizados para actualizar el controlador de eventos y cambios en el estado del switch. Existen cuatro tipos

2.7.2.1. Packet-in (Paquete entrante)

un mensaje Packet-in, es enviado al controlador; para todos los paquetes que no tienen una coincidencia «matching», con una entrada de la tabla del switch (flow-entry).

2.7.2.2. Flow-Removal (Eliminación de Flujo)

Cuando un flujo de entrada es adicionado al switch por medio de un mensaje Flow modify, tiene especificado el «idle_time_out», que es un valor de tiempo de inactividad indica cuando la entrada se debe quitar debido a la falta de actividad, y también un valor de tiempo de espera «hard_time_out» que indica cuando la entrada se debe quitar, independientemente de la actividad. El mensaje flow modify también especifica si el switch debe enviar un mensaje flow-removed al controlador cuando expire el flujo. Los mensajes flow modify que borran flujos también pueden causar mensajes flow removed.

2.7.2.3. Port-status (Estado del puerto)

Se espera que el switch envíe mensajes Port-status al controlador como cambios de estado en la configuración de puerto.

2.7.2.4. Error

El switch es capaz de notificar al controlador de problemas por medio de mensajes Error

2.7.3. Mensajes Simétricos

Estos mensajes son iniciados por el controlador o el switch, y enviados sin solicitación, en cualquier dirección.

2.7.3.1. Hello

Los mensajes Hello son intercambiados entre el switch y el controlador una vez la conexión arranque.

2.7.3.2. Echo (Eco)

Echo (Request/Reply), estos mensajes pueden ser enviados desde cualquier switch o controlador, y deben retornar un echo reply. Pueden ser utilizados para indicar la latencia, ancho de banda, y / o vida de la conexión que hay establecida entre el controlador y el switch.

2.7.3.3. Vendor (Proveedor)

Mensajes Vendor proporcionan una forma estándar para switches OpenFlow, con el fin de poder ofrecer funcionalidad adicional dentro del espacio de tipo de mensaje OpenFlow. Es un área de ensayo para las características destinadas a futuras revisiones OpenFlow.

2.8. Procesamiento de mensajes

Los switches deben procesar todos los mensajes recibidos de un controlador en su totalidad, posiblemente generando una respuesta. Si un switch no puede procesar completamente un mensaje recibido de un controlador, debe enviar un mensaje Error. Para los mensajes de salida de paquetes, el procesamiento completo del mensaje no garantiza que el paquete incluido realmente salga del switch. El paquete incluido se puede caer silenciosamente después del procesamiento de OpenFlow debido a la congestión en el switch. Además, los switches deben enviar al controlador todos los mensajes asíncronos generados por los cambios de estado de OpenFlow, como ser un flujo eliminado, el estado del puerto o mensajes Packet in, de modo que la vista del controlador del switch sea coherente con su estado real. Esos mensajes pueden filtrarse en función de la Configuración asíncrona (ver [7] 6.1.1). Además, las condiciones que desencadenan un cambio de estado de OpenFlow pueden filtrarse antes de causar dicho cambio. Por ejemplo, paquetes recibidos en puertos de datos que deben ser reenviados. El controlador puede caerse debido a la congestión o la política de QoS dentro del switch y no generar mensajes de paquetes. Estas caídas pueden ocurrir para paquetes con una acción de salida explícita al controlador. Estas caídas también pueden ocurrir cuando un paquete no coincide con ninguna entrada en una tabla y la acción predeterminada de esa tabla es enviar al controlador. Se recomienda la vigilancia de los paquetes destinados al controlador mediante acciones de QoS o limitación de velocidad, para evitar la denegación de servicio de la conexión del controlador, y está fuera del alcance de la presente especificación. Los controladores son libres de ignorar los mensajes que reciben, pero deben responder a los mensajes de eco para evitar que el switch finalice la conexión.

2.9. Ordenamiento de los mensajes

El orden de los mensajes, se pueden garantizar mediante el uso de mensajes de barrera (Barriers). En ausencia de mensajes de barrera, los switches pueden reordenar arbitrariamente los mensajes para maximizar el rendimiento; por lo tanto, los controladores no deben depender de un orden de procesamiento específico. En particular, las entradas de flujo pueden insertarse en tablas en un orden diferente al de los mensajes de modificación de flujo recibidos por el switch. Los mensajes no se deben

reordenar a través de un mensaje de barrera y el mensaje de barrera se debe procesar solo cuando se hayan procesado todos los mensajes anteriores. Más precisamente: Más precisamente:

- Los mensajes ante una barrera deben procesarse por completo, incluido el envío de cualquier respuesta o error resultante
- La barrera debe procesarse y enviar una respuesta de barrera.
- Los mensajes posteriores a la barrera pueden comenzar a procesarse. Si dos mensajes del controlador dependen uno del otro (por ejemplo, un mod de flujo se agrega con un paquete siguiente a OFPP_TABLE), deben estar separados por un mensaje de barrera

2.10. Canal OpenFlow

El canal OpenFlow se utiliza para intercambiar mensajes de OpenFlow entre un switch OpenFlow y un controlador OpenFlow. Un controlador típico de OpenFlow administra múltiples canales de OpenFlow, cada uno a un switch OpenFlow diferente. Un switch OpenFlow puede tener un canal OpenFlow para un solo controlador, o múltiples canales para mayor confiabilidad, cada uno para un controlador diferente (consulte [7] 6.3.4). Un controlador OpenFlow generalmente administra un switch OpenFlow de forma remota a través de una o más redes. La especificación de las redes utilizadas para los canales OpenFlow está fuera del alcance de la presente especificación. Puede ser una red dedicada separada, o el canal OpenFlow puede usar la red administrada por el switch OpenFlow (conexión de controlador en banda). El único requisito es que debe proporcionar conectividad TCP / IP. El canal OpenFlow se suele instanciar como una única conexión de red, utilizando TLS o TCP simple (consulte [7] 6.3.3). El canal OpenFlow puede estar compuesto de múltiples conexiones de red para explotar el paralelismo (ver [7] 6.3.5). El switch OpenFlow siempre inicia una conexión a un controlador OpenFlow (ver [7] 6.3.1)

2.10.1. Configuración de la conexión

El switch debe poder establecer comunicación con un controlador en una dirección IP configurable por el usuario (pero por lo demás fija), utilizando un puerto específico del usuario. Si el switch conoce la dirección IP del controlador, el switch inicia una conexión estándar TLS o TCP con el controlador. El tráfico hacia y desde el canal de OpenFlow no se ejecuta a través de la canalización de OpenFlow. Por lo tanto, el switch debe identificar el tráfico entrante como local antes de compararlo con las tablas de flujo. Cuando se establece por primera vez una conexión OpenFlow, cada lado de la conexión debe enviar inmediatamente un mensaje OFPT_HELLO con el campo de versión establecido en la versión más alta del protocolo OpenFlow admitida por el remitente (ver [7] A.1). Este mensaje de saludo puede incluir opcionalmente algunos elementos de OpenFlow para ayudar a la configuración de la conexión (ver [7] A.5.1). Al recibir este mensaje, el destinatario debe calcular la versión del protocolo OpenFlow que se utilizará. Si tanto el mensaje de saludo enviado como el mensaje de saludo recibido contenían un elemento de saludo OFPHET_VERSIONBITMAP, y si esos mapas de bits tienen algunos bits comunes establecidos, la versión negociada debe ser la versión más alta establecida en ambos mapas de bits. De lo contrario, la versión negociada debe ser la más pequeña del número de versión que se envió y la que se recibió en los campos de la versión. Si la versión negociada es compatible con el destinatario, entonces la conexión continúa. De lo contrario, el destinatario debe responder con un

mensaje OFPT_ERROR con un campo de tipo OFPET_HELLO_FAILED, un campo de código OFPHFC_INCOMPATIBLE, y opcionalmente una cadena ASCII que explica la situación en los datos, y luego terminar la conexión.

2.10.2. Interrupción de la Conexión

En el caso de que un switch pierda el contacto con todos los controladores, como resultado de los tiempos de espera de solicitud de eco, los tiempos de espera de la sesión TLS u otras desconexiones, el switch debe ingresar inmediatamente en "modo seguro de falla.º "modo independiente de falla", dependiendo de la implementación y configuración del switch. En el "modo seguro de falla", el único cambio en el comportamiento del switch, es que los paquetes y mensajes destinados a los controladores se eliminan. Las entradas de flujo deben continuar caducando de acuerdo con sus tiempos de espera en "modo seguro de falla". En "modo autónomo fallido", el switch procesa todos los paquetes utilizando el puerto reservado OFPP-NORMAL; en otras palabras, el switch actúa como un switch o enrutador Ethernet heredado. El "modo autónomo de falla"generalmente solo está disponible en los switches híbridos (ver [7] 5.1). Al conectarse nuevamente a un controlador, las entradas de flujo existentes permanecen. El controlador tiene la opción de eliminar todas las entradas de flujo, si lo desea. La primera vez que se inicia un switch, funcionará en el modo "modo seguro de falla.º "modo independiente de falla", hasta que se conecte con éxito a un controlador. La configuración del conjunto predeterminado de entradas de flujo que se utilizará en el inicio está fuera del alcance del protocolo OpenFlow.

2.10.3. Encriptación

El switch y el controlador pueden comunicarse a través de una conexión TLS. La conexión TLS es iniciada por el switch, el controlador se encuentra por defecto en el puerto TCP 6633. El switch y el controlador se autentican mutuamente intercambiando certificados firmados por una clave privada específica del sitio. Cada switch debe ser configurable por el usuario con un certificado para autenticar el controlador (certificado del controlador) y el otro para autenticarse en el controlador (certificado del switch). El switch y el controlador pueden comunicarse opcionalmente utilizando TCP simple. La conexión TCP es iniciada por el switch, el controlador se encuentra por defecto en el puerto TCP 6633. Cuando se usa TCP simple, se recomienda usar medidas de seguridad alternativas para evitar las escuchas ilegales, la suplantación del controlador u otros ataques en el canal OpenFlow.

2.10.4. Conexión para Controladores Múltiples

El switch puede establecer comunicación con un solo controlador, o puede establecer comunicación con múltiples controladores. Tener varios controladores mejora la confiabilidad, ya que el switch puede continuar operando en el modo OpenFlow si falla una conexión de controlador o controlador. El intercambio entre los controladores es administrado en su totalidad por los mismos controladores, lo que permite una rápida recuperación de fallas y también el balance de carga entre ellos. Los controladores coordinan la administración del switch entre ellos a través de mecanismos fuera del alcance de la especificación actual, y el objetivo de la funcionalidad de múltiples controladores es solo ayudar a sincronizar el intercambio o transferencias realizados entre ellos. La funcionalidad de controlador múltiple solo aborda la conmutación por error y el equilibrio de carga entre controladores, y no aborda la virtualización que pueda realizarse fuera del protocolo OpenFlow. Como se ve en en el punto2.7.1,

específicamente para los mensajes Role-Request: se prevén mecanismos para las funciones básicas mencionadas en el párrafo anterior. No obstante, los otros tipos de relaciones e intercambios que se puedan establecer entre múltiples controladores y/o dominios de control son materia de arquitecturas con interfaces este/oeste (eastbound/westbound) definidas, y estas interfaces como ya fue mencionado no están estandarizadas

2.10.5. Un comentario acerca del número de puerto para el controlador OpenFlow

La especificación OpenFlow 1.3.1, menciona que el puerto por defecto del controlador es 6633. Históricamente la Open Networking Foundation, utilizo el puerto TCP: 6633 para la conexión del controlador OpenFlow, a lo largo de las distintas versiones de su especificación, pero resulta que la empresa Cisco, con fecha 2012-06-11, registro para su producto Path Services Overlay [Cisco2], el puerto TCP 6633, ante el Service Name and Transport Protocol Port Number Registry [64] de la IANA [65]. Con respecto a esto hubo cierta polémica entre la Open Networking Foundation y la empresa Cisco. En el documento Mininet 2.2.2 Release Notes [66], se puede leer en el párrafo correspondiente a las correcciones de la versión; donde expresa literalmente: "The default OpenFlow port is now the official port of 6653 rather than the classic port of 6633, which was nastily co-opted by Cisco.Çualquier comentario o interpretación queda a cargo del lector ;)

Figura 2.22: Puerto OpenFlow

Efectivamente se pueden verificar ambos números en el registro según muestran los siguientes recortes de capturas de pantalla de la web del registro de la IANA citada anteriormente (ver Figura 2.22)

Dicho todo esto, se utiliza como puerto por defecto para protocolo OpenFlow el 6653 tanto para TCP como UDP, tal como quedo registrado por la Open Networking Foundation con fecha 2013-07-18. No obstante al utilizar plataformas de simulación como por ejemplo Mininet [67], como así también algunas versiones de controladores Open Source, se observa que se puede utilizar indistintamente 6633 o 6653 como puerto TCP/UDP.

2.11. Otros Protocolos Southbound Interface

Por la cantidad de actores y miembros de la Open Networking Foundation [61], el protocolo OpenFlow es prácticamente el estándar de facto para las comunicaciones en la Southbound Interface. No obstante, existe una variedad de otros protocolos que se implementan en esta interfaz, que cuentan con características interesantes que los hacen activos partícipes en la Southbound Interface de la arquitectura SDN. A continuación, mencionamos algunos de ellos.

2.11.1. Cisco OpFlex

Cisco OpFlex [12], es un protocolo en dirección sur en una red definida por software (SDN) diseñada para facilitar las comunicaciones entre el controlador SDN y la infraestructura (switches y enrutadores). El objetivo es crear un estándar que permita que las políticas se apliquen a través de switches / enrutadores físicos y virtuales en un entorno de múltiples proveedores. En la superficie, Cisco OpFlex se parece mucho a OpenFlow, un estándar abierto que permite al controlador SDN interactuar con la infraestructura; sin embargo, es bastante diferente en cuanto al alcance de sus capacidades. Mientras OpenFlow centraliza todas las funciones de control de red en el controlador SDN, Cisco OpFlex se enfoca principalmente en las políticas. Cisco cree que este enfoque eliminará la posibilidad de que el controlador se convierta en el cuello de botella de la red, que admita una mayor capacidad de recuperación, disponibilidad y escalabilidad, al llevar parte de la inteligencia a los dispositivos, utilizando los protocolos de red establecidos (ver Figura 2.23).

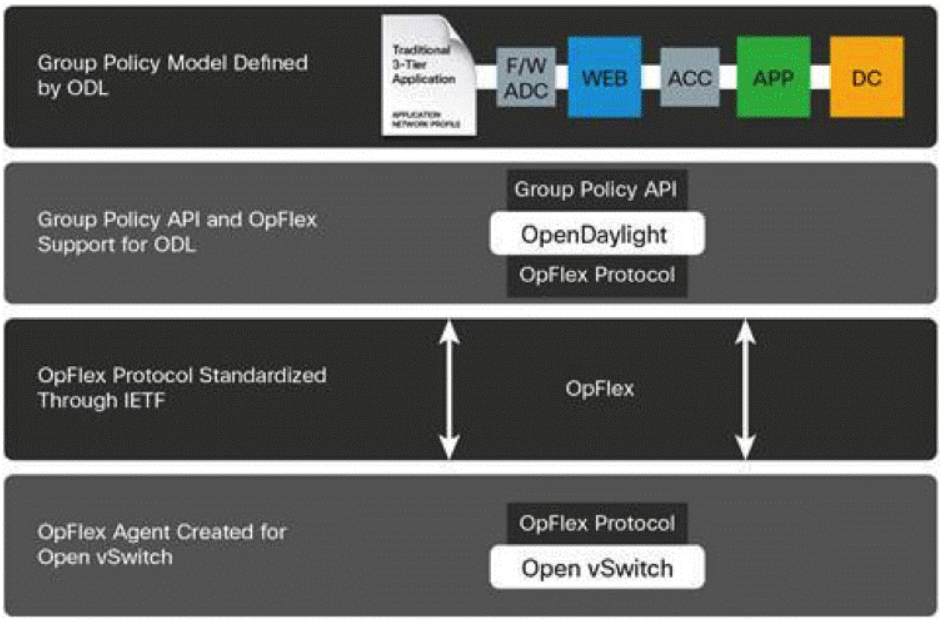

Figura 2.23: Estructura SDN con Cisco OpFlex [12]

Básicamente, las políticas se definen dentro de un repositorio de políticas lógico y centralizado en el controlador, y el protocolo OpFlex se usa para comunicar y aplicar esas políticas dentro de un conjunto de elementos de políticas distribuidas en los switches / enrutadores / etc. El protocolo permite la comunicación bidireccional de políticas, eventos, estadísticas e información de fallas, por lo que se pueden hacer ajustes para abordar los cambios en el entorno. Para trabajar, un agente debe estar integrado en los switches y enrutadores para admitir el protocolo Cisco OpFlex. Como resultado, Cisco está trabajando en un agente de código abierto de OpFlex que se puede usar en todas las plataformas. Las redes Microsoft, IBM, F5, Citrix, Red Hat, Canonical y AVI ya se han comprometido a integrar este agente en sus soluciones.

2.11.2. ForCES

ForCES [52] Proporciona un marco y define API / protocolos abiertos que separan claramente los planos de control y reenvío. Aunque se han desarrollado y / o propuesto muchos de estos protocolos / API (Ej. OpenFlow y RestAPI), la verdadera fortaleza de ForCES reside en su modelo que permite la descripción de la nueva funcionalidad de la ruta de datos sin cambiar el protocolo entre el plano de control y el de reenvío se puede ver en detalle en Figura 2.24.

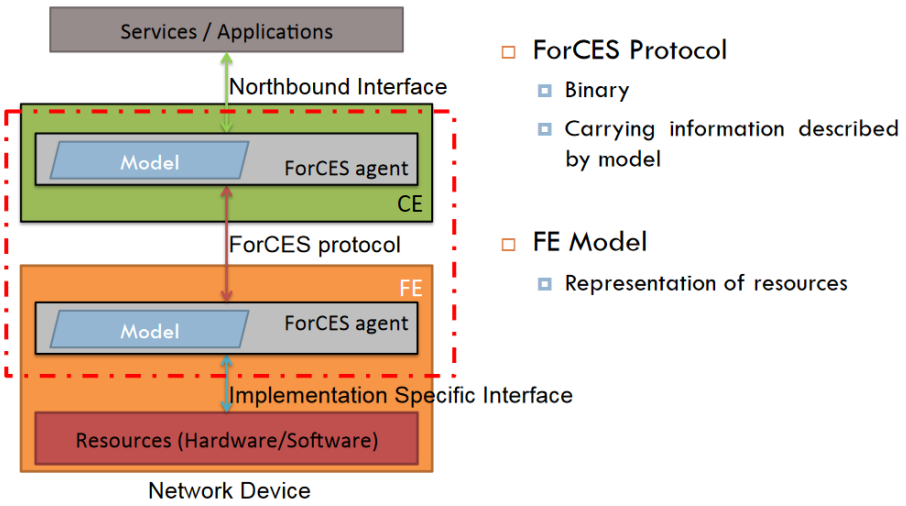

Figura 2.24: Visión Conceptual de ForCES [13]

En contraste, OpenFlow requiere la implementación del protocolo definido en el controlador y el plano de reenvío (por ejemplo, el agente de OpenFlow en el dispositivo de red). ForCES extiende la noción de arquitectura de enrutamiento elástico con la creencia de que el procesamiento de paquetes del plano de datos puede necesitar funcionalidades adicionales programadas en el futuro mientras que los switches se despliegan en la red. Esta abstracción lo diferencia de manera innata de otros protocolos / API SDN existentes, como OpenFlow.

Como muestra la Figura 2.25, ForCES sirve como un buen marco para desarrollar una arquitectura de enrutamiento elástica para dispositivos de red considerando los elementos de reenvío y elementos de control que forman parte de un sistema. De esa manera, el controlador centralizado y los dispositivos de red son capaces de intercambiar capacidades y ser parte de un sistema de arquitectura de enrutamiento elástico.

2.11.3. NETCONF

El protocolo de configuración de red (NETCONF) definido en la RFC 6241 [51] de la Internet Engineering Task Force (IETF) proporciona mecanismos para instalar, manipular y eliminar la configuración de dispositivos de red. Utiliza la codificación de marcado extensible de datos basada en lenguaje (XML) tanto para los datos, como para los mensajes de protocolo. Las operaciones del protocolo NETCONF son realizadas como llamadas a procedimiento remoto (RPC).

El protocolo básico de NETCONF se publicó oficialmente como RFC 4741 Protocolo de configuración de NETCONF a fines de 2006. El grupo de trabajo de IETF que produce la norma también produjo

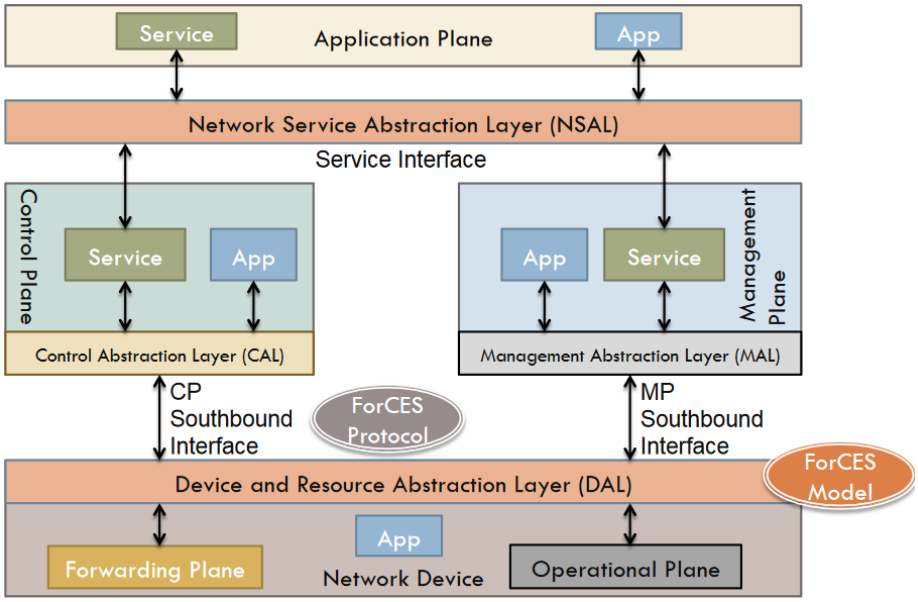

Figura 2.25: Arquitectura SDN implementada con ForCES [13]

RFC de apoyo para varias asignaciones de transporte, que incluyen:

- RFC 4742 usando el protocolo de configuración de NETCONF sobre Secure Shell (SSH)
- RFC 4743 usando NETCONF sobre el protocolo simple de acceso a objetos (Simple Object Access Protocol)
- RFC 5539 usando NETCONF sobre capa de transporte segura (TLS)

Las versiones anteriores se actualizaron en 2011 para convertirse en las siguientes:

- RFC 6241 deja obsoleta la RFC 4741 con un pequeño conjunto de cambios que incluye un ID de persistencia para confirmaciones
- RFC 6242 deja obsoleta la RFC 4742 e introduce, por ejemplo, un nuevo mecanismo de encuadre para abordar algunos problemas de seguridad potenciales con el diseño inicial

Las adiciones notables a la familia de RFC de NETCONF producidas por el grupo de trabajo son:

- RFC 5277 NETCONF Notificaciones de eventos que describen un mecanismo de notificación asíncrono que permite a los clientes suscribirse a secuencias de eventos con nombre
- RFC 6243 Capacidad predeterminada para NETCONF que describe una extensión del protocolo NETCONF que permite a los clientes identificar cómo procesa el servidor los valores predeterminados.

2.11.4. Palabras finales del Capítulo 2

Citando a [68], "La Open Networking Foundation (ONF) es una organización impulsada por los usuarios dedicada a la promoción y adopción de SDN a través del desarrollo de estándares abiertos. Esta organización, enfatiza la colaboración y el proceso abierto de desarrollo conducido por los usuarios, dando como resultado el mantenimiento del estándar abierto OpenFlow, el primer estándar SDN y un elemento vital para las arquitecturas abiertas de SDN. Hoy en día las comunidades de esta organización

siguen analizando los requerimientos de SDN y mejorando el estándar OpenFlow, para beneficiar SDN con nuevos estándares y suplir las necesidades de despliegues comerciales".

Es por este motivo, que elegimos trabajar en nuestro proyecto de investigación con OpenFlow; así también con otros elementos de la arquitectura SDN que pertenecen a proyectos colaborativos dentro del ámbito Open Source, pues es nuestra convicción, que esto además de facilitar el acceso y el intercambio de conocimientos, facilita la difusión y mejoramiento de todas las actividades desarrolladas en esta modalidad.El lector habrá notado que, si bien se menciona al Controlador SDN a lo largo de este capítulo, no se dieron mayores detalles acerca de su funcionalidad, ni variedad. Al tratarse de un componente clave dentro del plano operacional de control, nos referimos específicamente al Controlador SDN en el siguiente capítulo.

Capítulo 3

Controladores SDN

Como muestra la Figura 3.1 En el centro de la arquitectura de las SDN se encuentra el plano operacional denominado Plano de Control, allí se ubica o reside el controlador SDN, que es quien gestiona los flujos. El controlador equivale al sistema operativo de la red que controla todas las comunicaciones entre las aplicaciones y los dispositivos. El controlador SDN se encarga de traducir las necesidades o requisitos de plano operacional de las Aplicaciones a los elementos de red mediante las Interfaces norte SDN (Northbound Interfaces NBI), y de proporcionar información relevante a las aplicaciones SDN, pudiendo incluir estadísticas y eventos.

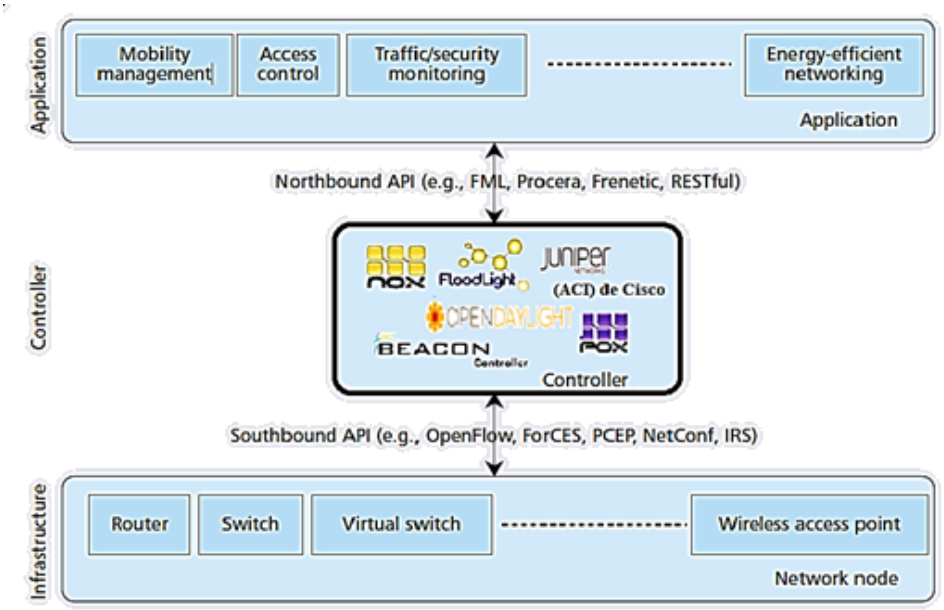

Figura 3.1: Arquitectura SDN, perspectiva desde el Controlador – Adaptado de [14]

En la parte inferior de la arquitectura, se encuentra la capa de Infraestructura, la que está compuesta por los elementos o dispositivos de red, que exponen sus capacidades a través de una interfaz de

control del plano operacional denominado Plano de Datos, también denominada interfaz "hacia el sur" (Southbound Interface), que permite la comunicación entre el controlador SDN y los dispositivos de red. La Southbound Interface permite como mínimo dos operaciones básicas:

- El control de programación de todas las operaciones de reenvío por parte del controlador (lectura y modificación de las tablas de reenvío de los dispositivos)
- El anuncio y aviso de capacidades, informes de estadísticas y la notificación de eventos de los dispositivos de red al controlador Las características y formatos de los mensajes intercambiados en esas operaciones los mostramos en el capítulo 2 sección 2.6.

La interfaz de control del plano de datos se formaliza, en forma casi universal, a través del protocolo OpenFlow estudiado en el capítulo 2, al estar desarrollado y promovido por Open Networking Foundation [46] se ha convertido en el protocolo estándar a utilizar para la conexión remota entre el controlador SDN y los dispositivos que controla en la infraestructura subyacente en todos los desarrollos y proyectos de controladores Open Source. Como resultado, la creación de una red SDN utilizando componentes de estas características, debe basarse en la selección de dispositivos y de software de control que soporten el estándar OpenFlow. Dado el permanente desarrollo de la tecnología de las SDN, y su característica emergente, resulta imposible enumerar la totalidad y variedad de controladores existentes, los aquí mencionados son algunos de los más frecuentes en variedad de estudios y publicaciones tanto académicas como comerciales. Por otra parte, no se consideran mayores detalles, pues están sobradamente documentados en las respectivas páginas de los proyectos y/o desarrolladores cuyo acceso se encuentra disponible en la sección Referencias.

3.1. Controladores Open Source

3.1.1. Controlador NOX

Fue uno de los primeros controladores desarrollados en el ámbito del proyecto de la Universidad de Standford [56], luego continua en Nicira Networks Copyright (C) 2008-2009, Stanford University Copyright (C) 2009-2010 International Computer Science Institute, UC Berkeley Copyright (C) 2009-2012. Se trata de una plataforma de control de red. Esta distribución incluye todo el software que necesita para compilar, instalar e implementar NOX en su red, así como el código fuente y las herramientas para permitirle desarrollar tus propias aplicaciones de NOX. El código fuente como así también los ejecutables están disponibles en [50]. Esta versión de NOX es una versión para desarrolladores. Está destinado a proporcionar una plataforma programática para controlar uno o más switches OpenFlow. NOX se puede ampliar tanto en C ++ como en una interfaz abstracta para OpenFlow. Esta distribución contiene un conjunto de aplicaciones de ejemplo y algunas bibliotecas integradas que proporcionan funciones de red útiles, como Seguimiento de host y enrutamiento.

3.1.2. Controlador POX

Es una plataforma de desarrollo de código abierto [69] para aplicaciones de control de redes definidas por software (SDN) basadas en Python, como los controladores OpenFlow SDN. POX, que permite un rápido desarrollo y creación de prototipos, se está utilizando más comúnmente que NOX, un proyecto hermano. El sitio web de NOXrepo [70] enumera las siguientes características de POX:

- Interfaz OpenFlow “Pythonic”.
- Componentes de muestra reutilizables para selección de ruta, descubrimiento de topología, etc.
- “Funciona en cualquier lugar”: se puede combinar con el tiempo de ejecución de Py sin instalación para una fácil implementación.
- Específicamente se dirige a Linux, Mac OS y Windows.
- Descubrimiento de topología.
- Admite las mismas herramientas de GUI y visualización que NOX.
- Se desempeña bien en comparación con las aplicaciones NOX escritas en Python.

El sitio web del proyecto dice que el objetivo final de POX es usarlo para crear ün controlador SDN arquetípico y moderno".

3.1.3. Controlador OpenDaylight

OpenDaylight anunció el lanzamiento de su plataforma de controlador SDN de código abierto, Hydrogen, en febrero de 2014. La edición base de Hydrogen incluye un controlador SDN multiprotocolo modular basado en OSGi (Open Service Gateway Initiative) [71], así como un complemento OpenFlow, un protocolo OpenFlow, Open vSwitch Database Management Protocol [72] y las herramientas YANG [73]. Con la introducción de un controlador SDN de código abierto, OpenDaylight declara que puede proporcionar un control centralizado para cualquier arquitectura SDN sin importar el proveedor (ver Figura 3.2). La página Wiki del controlador OpenDaylight incluye instrucciones sobre cómo instalar el controlador OpenDaylight, así como una guía del programador, preguntas frecuentes e información sobre la liberación de hidrógeno y helio. La última versión a la fecha en que esto se escribe es OpenDaylight Fluorine. Todos los detalles y todas las versiones pueden encontrarse en la web del proyecto [15].

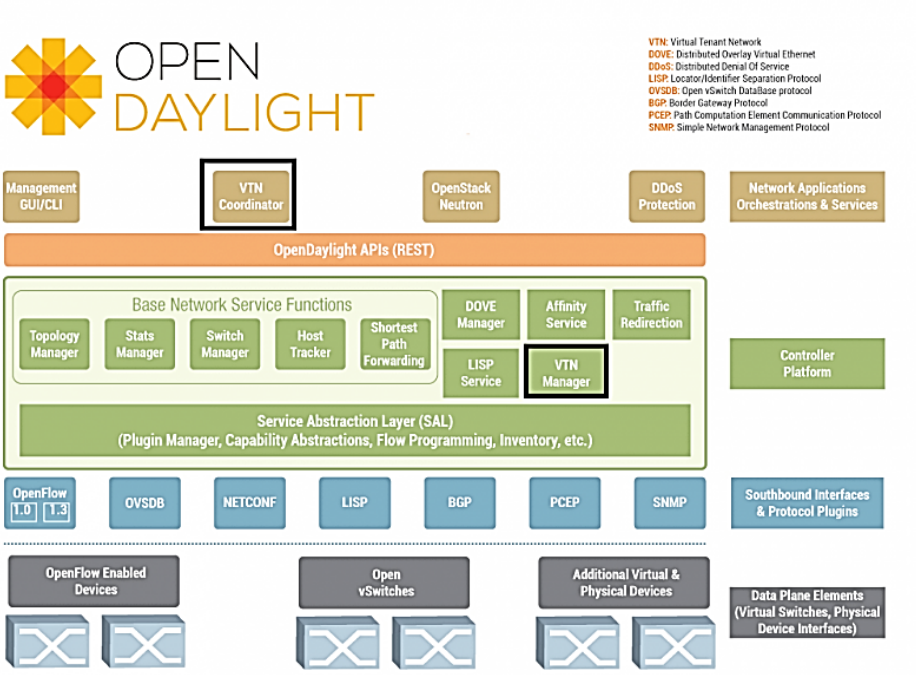

Figura 3.2: Estructura de OPENDAYLIGHT -Tomado de [15]

Posiblemente este controlador sea uno de los más ubicuos entre los controladores Open Source, por sus capacidades y características en permanente desarrollo, y la gran cantidad de actores en su comunidad de desarrollo.

3.1.4. Controlador OpenContrail

OpenContrail ofrece un controlador SDN como parte de su proyecto con licencia Apache 2.0 que se usa para habilitar la virtualización de la red. Aunque se considera abierto, OpenContrail proviene de Juniper Networks. El controlador funciona junto con los enrutadores virtuales que residen en hosts de hipervisor, un motor de análisis y las API publicadas hacia el norte (Northbound Interface). OpenContrail también puede actuar como una plataforma de red para infraestructura de nube. De acuerdo con el sitio Web del Proyecto [74], los aspectos clave de su sistema son la virtualización de la red, la programación y automatización de la red y el big data para la infraestructura. El código fuente de OpenContrail está alojado en múltiples repositorios de software como por ejemplo github.org (Juniper/contrail-controller) [75], con la funcionalidad central del sistema que se encuentra en el repositorio de controlador de contrail. Las instrucciones adicionales del código fuente también están disponibles.

3.1.5. Controlador Floodlight

Floodlight es un controlador OpenFlow basado en Java que también es de clase empresarial y tiene licencia Apache. Es parte de una colección de proyectos de código abierto realizados por Big Switch. El controlador es compatible con una variedad de switches virtuales y físicos de OpenFlow y puede manejar redes mixtas de OpenFlow y no OpenFlow. El controlador también incluye soporte para la plataforma de organización en la nube OpenStack. Floodlight ya se ha utilizado en varias aplicaciones, incluido el complemento Quantum de OpenStack y el switch virtual de Floodlight. El controlador está disponible para descargar en el sitio web del proyecto [76].

3.1.6. Controlador Ryu OpenFlow

Ryu es un framework SDN que ofrece componentes de software utilizados en aplicaciones SDN. Permite a los desarrolladores crear nuevas aplicaciones de gestión y control de red. Ryu es compatible con varios protocolos para administrar dispositivos de red, incluidos OpenFlow, Netconf y OF-config. La documentación está disponible para obtener más información sobre el sistema operativo de la red Ryu y está disponible para descargar en el sitio del proyecto. [77]

3.1.7. Controlador FlowVisor OpenFlow

FlowVisor es un controlador OpenFlow de propósito especial que actúa como un intermediario entre los switches OpenFlow y varios controladores OpenFlow. El controlador permite la virtualización de la red dividiendo una red física en múltiples redes lógicas. El controlador se asegura de que cada uno de los controladores toque solo los switches y los recursos asignados. También particiona el ancho de banda y los recursos de la tabla de flujo en cada switch y asigna particiones a controladores individuales. El controlador FlowVisor está disponible para descargar con instrucciones en su sitio de GitHub. [78]

3.2. Controladores Propietarios

3.2.1. VMware

VMware [79] proporciona una solución de orquestación de centro de datos con un controlador SDN [80] y una implementación de agente que se ha convertido en un estándar de facto. VMware fue una de las empresas de génesis para la computación en la nube, fundada en 1998. VMware ofrece un conjunto de aplicaciones centradas en el centro de datos creadas alrededor del hipervisor ESX (ESXi para la versión 5.0 y posteriores) y el switch del hipervisor, el switch distribuido vSphere [VDS]).

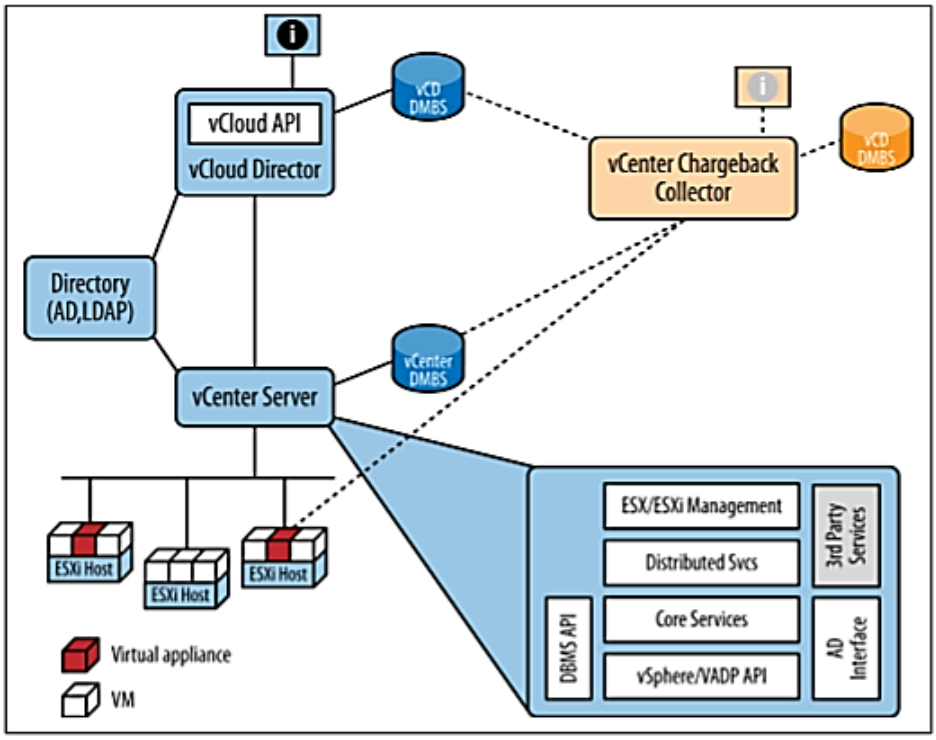

Figura 3.3: Relaciones de productos VMware (con vCenter Chargeback Collector como ejemplo de cómo se conectaría Operations Management Suite) – Tomado de [1]

La Figura 3.3 muestra un resumen de las relaciones de productos de VMware. vSphere introdujo el hipervisor ESXi (con la versión 5.x) para reemplazar el hipervisor ESX más antiguo, haciéndolo más liviano / más pequeño (de acuerdo con los pronunciamientos de marketing; ESXi es el 5 % del tamaño de ESX) y el sistema operativo independiente. El cambio también agrega una interfaz web a las opciones de administración de ESX existentes de CLI, API de cliente y visualización de vCenter. También eliminó una VM invitada requerida (es decir, una VM invitada por host) para una consola de servicio para la administración local.

3.2.2. Juniper

NorthStar Controller [81]: Se trata de un conjunto de herramientas potente y flexible. El primer controlador de redes definidas por software (SDN) de la industria para la optimización del tráfico proporciona visibilidad granular y control sobre los flujos de IP / MPLS en grandes redes de proveedores

de servicios y empresas. Puede usarlo para optimizar su infraestructura al monitorear los datos de telemetría. Las herramientas de planificación lo ayudan a crear rutas de enrutamiento explícitas dinámicamente utilizando una vista global basada en restricciones definidas por el usuario. El controlador WAN SDN automatiza la creación de rutas de ingeniería de tráfico a través de la red, aumentando la utilización de la red y permitiendo una experiencia de red personalizada y programable. A través de la potencia de Junos OS, los algoritmos de optimización WANDL y la abstracción del transporte, NorthStar Controller permite diseños eficientes. NorthStar reúne los análisis de Junos IP / MPLS y libera nuevos niveles de control y visibilidad que le ayudan a evitar el exceso de provisiones costosas.

3.2.3. Cisco

La Infraestructura centrada en aplicaciones (ACI) de Cisco [82]: En su web describe una plataforma basada en SDN [82] de la siguiente manera: Nuestros productos y soluciones se basan en una arquitectura de red abierta y son compatibles con su entorno de múltiples proveedores. Construye sobre ellos e innova cuando lo necesites. Las principales tecnologías incluyen:

- Análisis en tiempo real
- Modelo de telemetría conducida
- Orquestación Multi-vendedor

Juntos, estos pueden ayudar a transformar sus operaciones de red y su negocio. En particular el componente Application Policy Infrastructure Controller (APIC), representa al controlador dentro del esquema ACI.

3.2.4. Hewlett Packard

La compañía Hewlett Packard puso a disposición de la comunidad un controlador desarrollado por ellos con una versión libre y otra con desarrollos propietarios [83] su descripción es la siguiente: El software del controlador SDN de las redes de aplicaciones virtuales VAN de HP, proporciona un punto de control unificado en una red habilitada para OpenFlow, lo que simplifica la administración, el aprovisionamiento y la orquestación. Esto permite la entrega de una nueva generación de servicios de red basados en aplicaciones. También proporciona interfaces abiertas de programas de aplicación (API) para permitir a los desarrolladores externos ofrecer soluciones innovadoras para vincular dinámicamente los requisitos empresariales a la infraestructura de red a través de programas Java personalizados o interfaces de control REST [84]de propósito general. El controlador VAN SDN está diseñado para funcionar en entornos de campus, centros de datos o proveedores de servicios. Sus principales características son:

- Plataforma de clase empresarial para la entrega de una amplia gama de innovaciones de red
- Cumple con los protocolos OpenFlow 1.0 y 1.3
- Compatibilidad con más de 50 modelos de switches HP habilitados para OpenFlow
- API abiertas para permitir el desarrollo de aplicaciones SDN de terceros
- Arquitectura de controlador extensible, escalable y resistente

Hasta aquí se hace un repaso de los controladores SDN y sus características, esto sumado a los capítulos anteriores brinda una base suficiente para abordar el siguiente capítulo que trata de la experimentación y simulación en entornos SDN.

Capítulo 4

Simulación en entornos SDN

En los anteriores capítulos, se estudia y explica, desde la idea germinal hasta los fundamentos y especificaciones de las redes SDN, proporcionando el conocimiento básico necesario para encarar las simulaciones. En este capítulo se trata el contexto de la simulación y su importancia tanto en el aspecto académico, como en el ámbito de producción. Aspectos importantes para señalar:

- El desarrollo del capítulo, incluye la selección de aquellos recursos que en la opinión del grupo de investigación cumplen mejor con el criterio de priorizar en lo posible recursos Open Source [85], GNU (General Public License) [86], GNU versión 2 [87] y Software Libre [88], en la medida que cumplan con los estándares de calidad y funcionalidad profesional requeridos, y también que estén avalados por la industria, comunidades académicas y ubicuidad en el campo de la simulación.
- Las simulaciones desarrolladas en el capítulo, además de explicar los recursos utilizados, se exponen de manera "tutorial", poniendo énfasis en los detalles y resultados para facilitar su reproducción. Mostrando y explicando: El escenario utilizado; ¿Cómo configurar?, "¿Cómo hacer?, que precauciones tener, etc.
- Los ejemplos incluidos, son los que consideramos más representativos, representan un resumen de nuestro trabajo de experimentación, y tienen (al menos en nuestra intención) el objetivo de disminuir al lector, el tiempo que insume la curva de aprendizaje, tanto de las herramientas como su aplicación

4.1. El contexto de la simulación en redes de nueva generación

A medida que aumentan los servicios y el desarrollo de software, las exigencias de usuarios para contar con disponibilidad, garantías, calidad de servicio y seguridad de la información son cada vez mayores. Las arquitecturas en muchas de las redes existentes no fueron diseñadas para satisfacer la velocidad con la que aumentan esas demandas, provocando que los operadores de redes, organizaciones y empresas, en no pocas oportunidades, vean limitadas sus posibilidades de crear, expandir o modificar modelos de negocio y servicios. Por este motivo, en comunidades académicas, departamentos de investigación

y desarrollo (RD), empresas y de organizaciones, se impulsa a examinar las arquitecturas actuales, buscando responder con nuevas técnicas y tecnologías la veloz y creciente demanda. Precisamente, son los límites los que impulsan la creación de las redes de nueva generación La llegada de la virtualización ha flexibilizado la definición y el uso de sus recursos, permitiendo definir y modificar en tiempo real, a nivel de software, una infraestructura completa basada en perfiles de aplicaciones, en necesidades de rendimiento, etc. El análisis de infraestructura de redes y DataCenters orientados a servicios se vuelve un elemento crítico en las empresas de telecomunicaciones. Aquellos dispositivos de red, que son compatibles con la definición de switch OpenFlow-híbrido (ver punto 2.4.8 del Cap. 2), permiten la conmutación basada en OpenFlow, tanto como la operación normal, lo que permite a las organizaciones introducir progresivamente esta tecnología, incluso en entornos de red de múltiples proveedores. A pesar de la flexibilidad mencionada, resulta claro, que trabajar experimentando en entornos de producción, implica asumir muchos riesgos en cuanto a las políticas de seguridad de los operadores, como así también, una serie de limitaciones operativas. La gran ventaja de la separación basada en software de plano de control / infraestructura, permite simulaciones que a su vez pueden dar lugar a desarrollos o soluciones trasladables en forma directa a entornos de producción. Expresado esto, queda muy clara la importancia de la experimentación / simulación y el estudio de recursos específicos para poder llevarlas a cabo.

4.2. Laboratorio y escenario de simulación

La simulación se puede realizar en abarcando la totalidad de los planos operativos de la arquitectura SDN, o también se puede realizar a nivel de un plano en particular, montando los recursos apropiados en una computadora de uso general. Las posibilidades abarcan analizar el comportamiento de la red estudiada, y lo que no es menos importante desde el punto de vista de la investigación, visualizar y comprender la interacción entre los dispositivos de networking y protocolos utilizados. Para realizar las distintas experiencias de simulación y/o emulación de infraestructuras y topologías de redes, se procedió a la construcción del “Laboratorio de Experimentación SDN”, como decidimos llamar en forma afectuosa, y tal vez muy formalmente... a la computadora que se utiliza en todas las experiencias que se describen en este capítulo. Más allá de la humorada, es necesario destacar que, con una inversión, consistente en el costo de una pc o estación de trabajo, con hardware que cubra los requisitos del sistema operativo y los distintos recursos de software empleados (los que en gran mayoría son Open Source o licencia GNU de costo “cero”), el grado de experimentación, complejidad de simulación y resultados alcanzables es notablemente alto; y esto en gran medida gracias a la calidad del enorme trabajo, que diariamente aportan las distintas comunidades de desarrolladores en los proyectos Open Source mencionados en este libro. Habiendo expresado esto, se recomendamos (en base a nuestra experiencia), que la computadora disponga de un procesador como mínimo de cuatro núcleos y memoria RAM mínimo 16 GB estas características agilizan notablemente las experiencias de simulación, aunque en la documentación del software empleado (leer las referencias), se puede ver que es posible trabajar con una exigencia en cuanto a hardware de base, bastante menor que la mencionada.

4.2.1. Estructura de simulación / emulación

La Figura 4.1 muestra la disposición básica del equipamiento y recursos utilizados en las simulaciones descriptas en el presente capítulo.

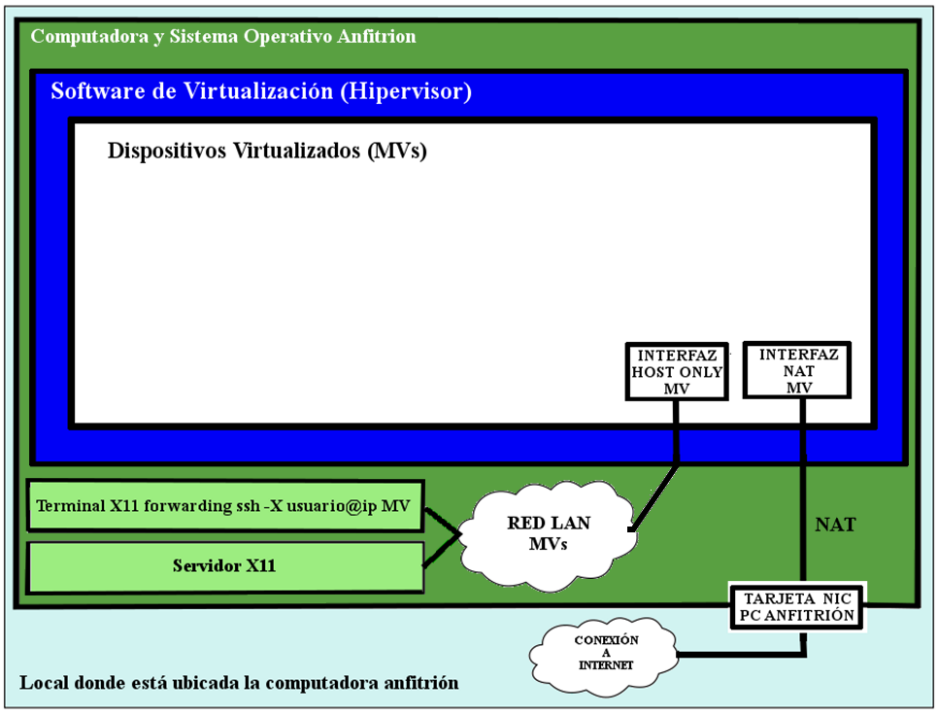

Figura 4.1: Esquemas de bloques de los componentes del Laboratorio SDN

4.3. Descripción de la computadora

4.3.1. Hardware

• Placa base: fabricante: ASUSTeK COMPUTER INC. - producto: PRIME B250M-A • BIOS: fabricante: American Megatrends Inc. Versión: 1001 date: 12/11/2017- tamaño: 64KiB - capacidad: 15MiB. • Microprocesador: producto: Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz - fabricante: Intel Corp. - versión: Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz configuración: cores=4 enabledcores=4 threads=8 - ranura: LGA1151 – 800MHz - capacidad: 4200MHz - 64 bits - reloj: 100MHz • Memoria de sistema: tamaño: 16GiB • Bank:0: descripción: DIMM DDR4 Síncrono Unbuffered (Unregistered) 2133 MHz (0,5 ns) - producto: 9905625-062.A00G - fabricante: Kingston - ranura: ChannelA-DIMM1 - tamaño: 8GiB - 64 bits - reloj: 2133MHz (0.5ns). • Bank 2: descripción: DIMM DDR4 Síncrono Unbuffered (Unregistered) 2133 MHz (0,5 ns) - producto: 9905625-062.A00G - fabricante: Kingston - ranura: ChannelB-DIMM1- tamaño: 8GiB - 64 bits - reloj: 2133MHz (0.5ns). • Disco Rígido: fabricante: Western Digitaldescripción: ATA Disk - producto: WDC WD10EZEX-08W - tamaño: 931GiB (1TB) - descripción: partición EXT4 - fabricante: Linux - tamaño: 465GiB. • Placa de Red: descripción: Ethernet interface - producto: RTL8111/8168/8411 PCI Express Gigabit Ethernet Controller • Multimedia: descripción: Audio device- producto: 200 Series PCH HD Audio - fabricante: Intel Corporation - 64 bits reloj: 33MHz.

4.3.2. Software

• El sistema operativo anfitrión es Ubuntu 18.04.1 LTS [89] • El entorno de virtualización es Virtual VirtualBox (release 5.2.2) [90] • El entorno de emulación Mininet (release 2.2.2) [66] •

Analizador de protocolos Wireshark [91]

4.4. Entorno de emulación Mininet

Mininet [67] es un entorno de emulación que crea una red virtual realista, en segundos, con un solo comando, ejecutando kernel real, switches y código de aplicación, en una sola máquina (VM, cloud o nativa):

- Debido a que puede interactuar fácilmente con su red utilizando la CLI de Mininet (y la API), personalizarla, compartirla con otros o desplegarla en hardware real, Mininet es útil para el desarrollo, la enseñanza y la investigación.
- Mininet es también una excelente forma de desarrollar, compartir y experimentar con sistemas OpenFlow y Software Defined Networking (SDN).
- Mininet se desarrolla activamente y es apoyado, y se libera bajo una licencia permisible BSD Open Source.
- Mininet funciona únicamente sobre Linux, concretamente sólo en aquellos que usan los kernels más nuevos, en concreto esos que traen con el soporte para lo que se conoce como "network namespaces". Se recomienda que el sistema Linux que se use tenga un kernel superior al 2.6.33. Además, Mininet garantiza su buen funcionamiento si se usa sobre una distribución Ubuntu, aunque poco a poco va extendiéndose. Debido a esto; se recomienda trabajar con máquinas virtuales, entre otras razones por su facilidad de transporte y capacidad de reunir una configuración de entorno compacta en un solo archivo.

Ventajas de Mininet

- Puede crear topologías personalizadas: un solo switch, grandes topologías tipo Internet, una red troncal, un centro de datos, o cualquier otra cosa de redes.
- Es rápido, la puesta en marcha de una red simple tarda sólo unos segundos. Esto significa que el bucle de gestión edit-debug puede ser muy rápido.
- Ejecutar programas reales: cualquier cosa que se ejecute en Linux está disponible para que funcione, desde los servidores de Internet hasta las herramientas de TCP para el monitoreo de paquetes.
- Personalizar el reenvío de paquetes: los switch de Mininet son programables usando el protocolo OpenFlow, lo que significa que los diseños de red se pueden transferir fácilmente a los switches OpenFlow.
- Puede ejecutar Mininet en su computadora portátil, en un servidor, en una máquina virtual, en una máquina Linux nativo (se encuentra en los repositorios), o en la nube (por ejemplo, Amazon EC2)
- Puede compartir y replicar los resultados: cualquiera con una computadora puede ejecutar el código una vez que se han instalado los paquetes necesarios para su funcionamiento.
- Se puede crear y ejecutar experimentos Mininet escribiendo simples (o complejos de ser necesario) scripts de Python. (cabe destacar sobre este punto, que Mininet tiene una aplicación de interfaz gráfica llamada "Miniedit", que permite armar cualquier topología arbitraria y genera la script correspondiente en Phyton, lo que resulta extremadamente conveniente para quienes no manejan este lenguaje de programación, facilitándole introducir cambios o ajustes a la misma)

Limitaciones de Mininet

- Correr en un solo sistema es conveniente, pero también impone límites de recursos: si su servidor tiene 3 GHz de CPU y puede cambiar alrededor de 3 Gbps de tráfico simulado, tendrán que ser

equilibradas y compartidas entre los hosts virtuales y switches.

- Mininet utiliza un solo núcleo de Linux para todos los hosts virtuales, lo que significa que no se puede ejecutar el software que depende de BSD, Windows, u otros núcleos del sistema operativo. (Aunque se puede conectar máquinas virtuales a Mininet.)
- Todos los hosts Mininet comparten el sistema de archivos y el espacio PID, lo que significa que es posible que tenga que tener cuidado si se está ejecutando demonios que requieren de configuración en /etc, y hay que tener cuidado de que no se mata a los procesos erróneos por error. (Tenga en cuenta el ejemplo bind.py que demuestra cómo tener directorios privados por el huésped.)
- A diferencia de un simulador, Mininet no tiene una fuerte noción de tiempo virtual, lo que significa que las medidas de temporalización se basarán en tiempo real, y que los resultados serán en tiempo real (por ejemplo, redes a 100 Gbps, requieren una cuidadosa configuración para ser emuladas).
- La funcionalidad de los switches depende del controlador y versión OpenFlow utilizados.

La forma más fácil de empezar es descargando una máquina virtual Mininet / Ubuntu pre-empaquetada. La MV está disponible en la página de Mininet (en nuestro caso se utiliza la versión 2.2.2) Esta MV incluye Mininet en sí, todos los binarios y herramientas de OpenFlow preinstalados, ajustes a la configuración del kernel para soportar redes más grandes de Mininet, y también Wireshark con los disectores OpenFlow, un controlador de referencia y el controlador POX [69].

- Opción 1: Instalación de Mininet VM (fácil, recomendado)
- Opción 2: Instalación nativa de origen
- Opción 3: Instalación de paquetes
- Opción 4. Actualización de una instalación de Mininet existente

El párrafo anterior hace una muy somera descripción de este poderoso entorno de emulación, no obstante, la página del proyecto ofrece excelentes reseñas y descripciones. A continuación, se hacen algunas consideraciones que nos parecen útiles conocer antes de arrancar de lleno con ejemplos de simulaciones a manera de tutorial.

4.4.1. Utilizando VirtualBox para la VM Mininet

Por razones de practicidad y además de tratarse de un excelente software de uso gratuito se utiliza Virtual Box corriendo una MV Mininet, para nuestras simulaciones con Mininet. VirtualBox es un potente producto de virtualización para plataformas x86 y AMD64 / Intel64, para empresas y para uso doméstico. VirtualBox no solo es un producto extremadamente rico en funciones y alto rendimiento para clientes empresariales, sino que también es la única solución profesional que está disponible gratuitamente como software de código abierto según los términos de la Licencia Pública General de GNU (GPL) versión 2.

4.4.1.1. Tips - Consideraciones acerca de la configuración de la MV Mininet en VirtualBox

- Primer paso importar a VirtualBox la MV Mininet que se obtuvo de la web del proyecto [67]
- Una vez que tenemos importada la MV Mininet, se recomienda verificar que estén activado los adaptadores NAT y Host-Only y que ambos sean del mismo tipo, por Ej: Intel PRO/1000MT Desktop, (esto lo sugerimos pues en ocasiones siendo de distintos tipos, experimentamos inconvenientes como ser que no funcione dhcp en la interfaz NAT, y algunas otras fallas en la configuración. Pero tomando esta precaución no tuvimos ningún inconveniente).

- Se recomienda regenerar las claves predeterminadas que se utilicen en las máquinas virtuales Mininet (o derivadas) utilizando los siguientes comandos: o *sudo rm -f /etc/ssh/*key* o sudo /usr/sbin/dpkg-reconfigure openssh-server o sudo service ssh restart*
- Modificar en la MV Mininet, el archivo interfaces como muestra la Figura 4.2. Hay que editar el archivo interfaces con cualquier editor de su preferencia. En nuestro caso utilizamos "nano", entonces: sudo nano /etc/network/interfacesFigura 4 2. Haciendo esto, las interfaces quedan configuradas en forma permanente y ya no debemos preocuparnos más por configurar las direcciones cada vez que iniciemos la MV (en este caso la red de máquinas virtuales está en 192.168.56.0 / 24)
- Cuando se inicia la MV Mininet en el entorno VirtualBox sugerimos hacerlo con la opción "inicio sin pantalla", esto evitará que tengamos inútilmente abierta otra ventana (que no se utiliza), pues como veremos durante el desarrollo de las experiencias, toda la comunicación con la máquina virtual la hacemos mediante terminales y ssh.

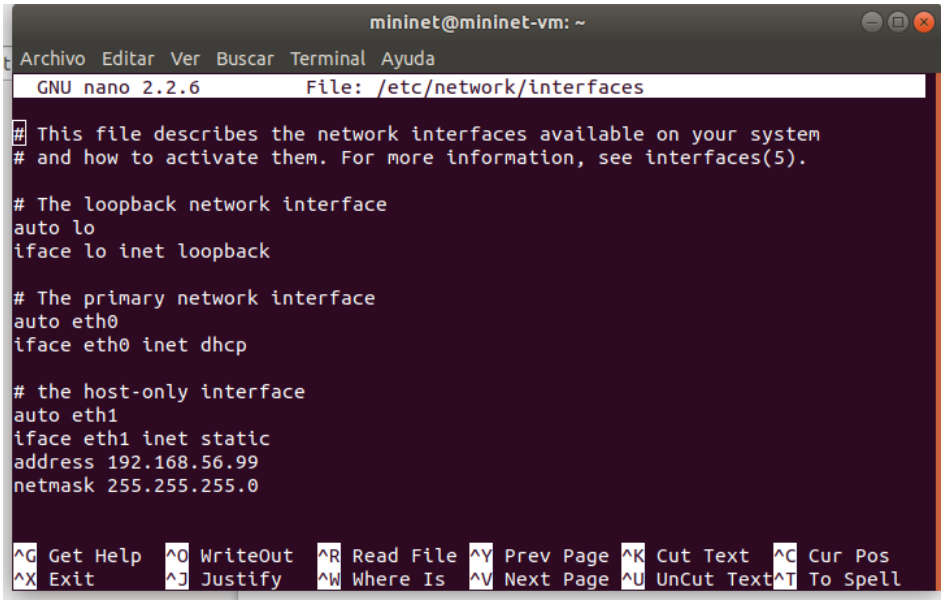

Figura 4.2: Editando /etc/network/interfaces

Ya estamos en condiciones de empezar a utilizar Mininet, a continuación, se expone una breve introducción a los comandos básicos de Mininet, que sugerimos practicar.

4.4.2. Mininet: Interactuando con hosts y switches

La topología predeterminada es la topología "minimal" (Figura 4.3), que incluye un switch de kernel OpenFlow conectado a dos hosts, más el controlador de referencia OpenFlow. Se la utiliza con el siguiente comando: • $sudo mn –topo=minimal

Esta topología básica es muy útil para quienes quieran hacer un estudio del comportamiento de OpenFlow y visualizar los detalles de los mensajes intercambiados con wireshark, encabezados, tipos de mensaje, valores de los campos, etc. Permite de manera inmediata y sencilla empezar a experimentar los comandos de control del switch, administrar y visualizar los flujos y observar el comportamiento del tráfico en un entorno SDN completo, utilizando el CLI de Mininet. No obstante, como iremos viendo,

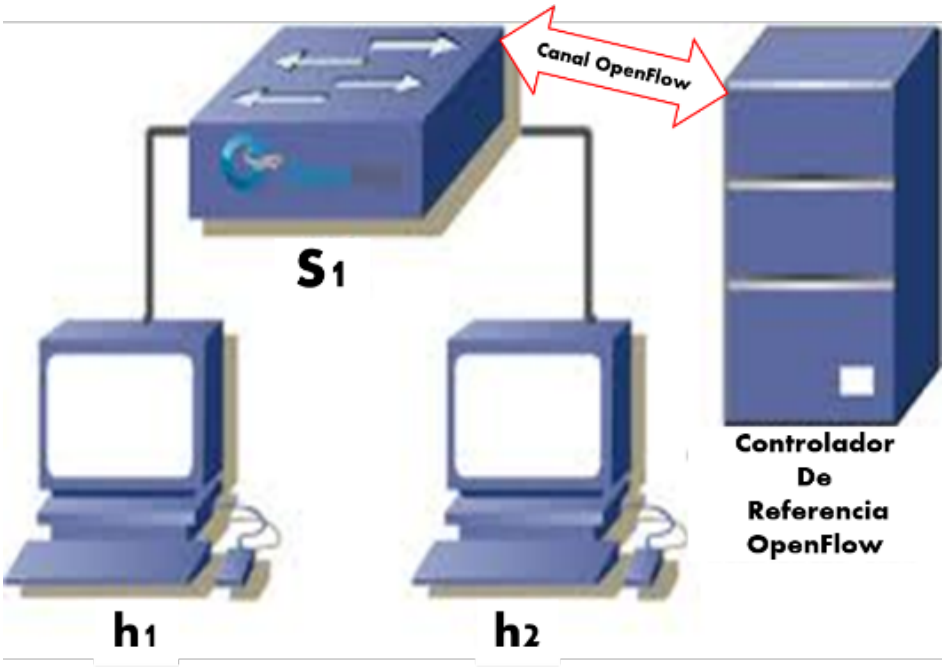

Figura 4.3: Topología “minimal” de Mininet

con la opción “–topo”, se pueden crear topologías personalizadas tales como: un solo switch, varios switches, redes de campus, grandes topologías tipo Internet, una red troncal, un centro de datos, o casi cualquier topología arbitrariamente compleja. Algunos comandos iniciales:

- Display Mininet CLI commands: Mininet>help
- Display nodes: Mininet>nodes
- Display links: Mininet>net
- Dump information about all nodes: Mininet>dump
-

Una nota (quizás obvia, pero importante tener en cuenta):

- $ precede a los comandos de Linux que se deben escribir en el prompt de Shell (terminal de usuario) en la MV Mininet
- Mininet>precede los comandos en el CLI de Mininet, dentro del entorno de emulación
- # precede a los comandos de Linux que se escriben en un prompt de root (terminal de root) en la MV Mininet

Se hace esta aclaración porque es común al efectuar simulaciones, trabajar con muchas terminales abiertas simultáneamente en pantalla y es fácil confundir cual se está utilizando al momento de introducir un comando.

4.4.3. Mininet: Ejemplos comandos de creación de topologías básicas

4.4.3.1. Opción "–topo"

La Figura 4.4 muestra como con una sencilla línea de comando se pueden crear distintas topologías en Mininet.

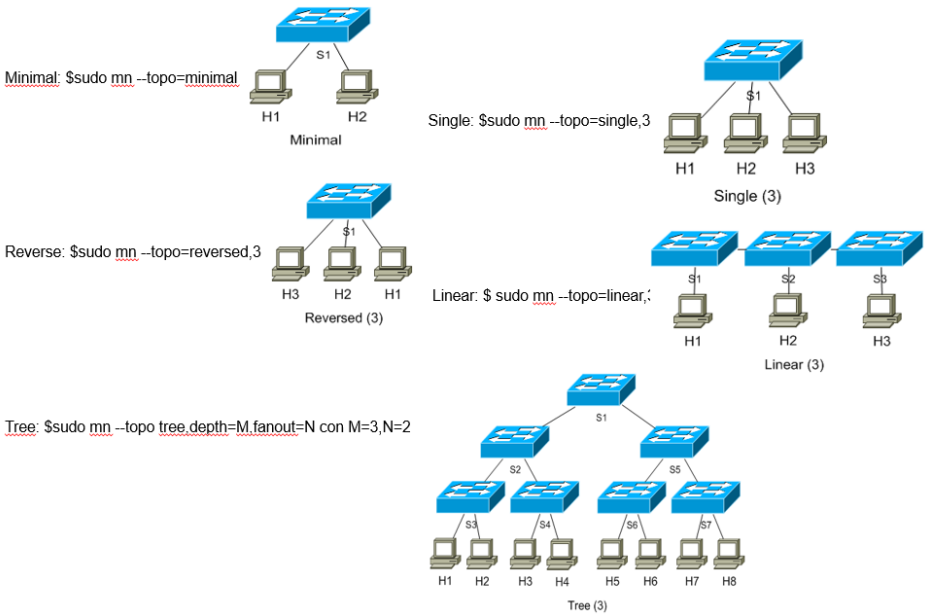

Figura 4.4: Creación de topologías con Mininet

Torus: genera una topología en forma de malla X x Y donde cada switch se conecta con sus vecinos más cercanos y los switches del borde se conectan con el opuesto, estas topologías son usadas normalmente en la interconexión de sistemas de computación paralela, Mininet solo soporta topologías del toro 2D, esta topología tiene loops y podría no funcionar con dispositivos incompatibles con el protocolo Spanning tree, por lo tanto, es recomendado el uso de esta topología con switches lxbr (Linux Bridge); compatibles con el protocolo STP. Ejemplo: "sudo mn –topo torus,x=3,y=3 –switch lxbr,stp=1 –test pingall". Recibe como argumento x=[N], y=[N] (ver Figura 4.5)

4.4.3.2. Opción "–mac"

Por defecto, los hosts comienzan con direcciones MAC asignadas al azar. Esto puede hacer que la depuración sea difícil, ya que cada vez que se crea Mininet, las direcciones MAC cambian, por lo que es difícil correlacionar el tráfico de control con hosts específicos. La opción –mac es muy útil y configura las direcciones IP y MAC del host en ID pequeñas, únicas y fáciles de leer; por ejemplo, el host1 IP: 10.0.0.1; MAC: 00:00:00:00:00:01 y así sucesivamente, la relación entre el host y sus respectivas direcciones es que termina con el número que identifica al host, esto facilita mucho la identificación del host en el estudio del tráfico La Figura 4.6 muestra configuración de interfaces sin la opción "–mac" y la Figura 4.7 con la opción "–mac"

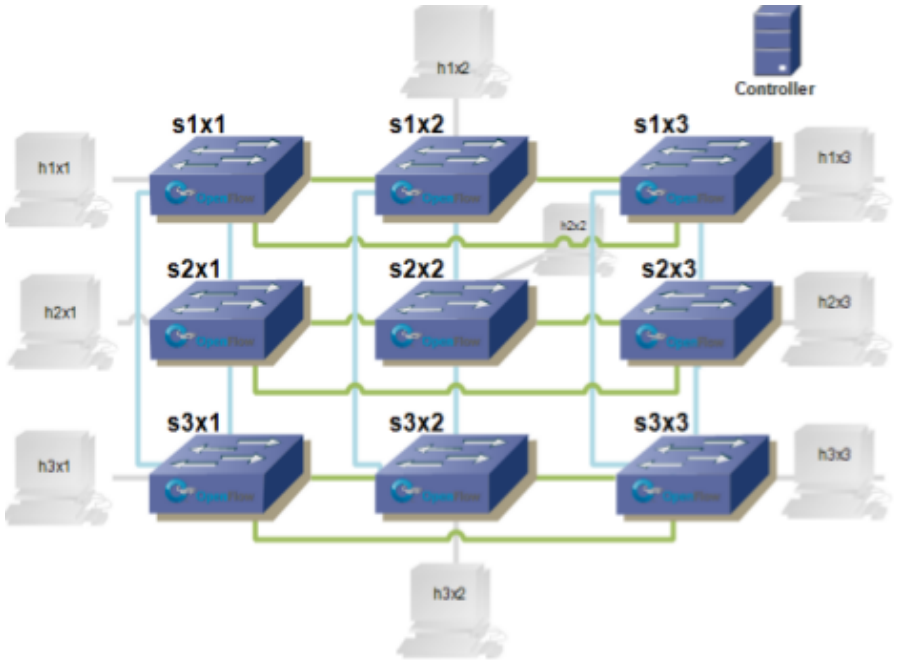

Figura 4.5: Topología “Torus” en Mininet

```
$ sudo mn
...
mininet> h1 ifconfig
h1-eth0  Link encap:Ethernet  HWaddr f6:9d:5a:7f:41:42
         inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:6 errors:0 dropped:0 overruns:0 frame:0
         TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:392 (392.0 B)  TX bytes:392 (392.0 B)
mininet> exit
```

Figura 4.6: Configurando Interfaces sin la opción “- -mac”

```
$ sudo mn --mac
...
mininet> h1 ifconfig
h1-eth0  Link encap:Ethernet  HWaddr 00:00:00:00:00:01
         inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:0 errors:0 dropped:0 overruns:0 frame:0
         TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
mininet> exit
```

Figura 4.7: Configurando Interfaces con la opción “- -mac”

4.4.3.3. Opción "–link": Parámetros de los enlaces

- Enlaces emulados

La velocidad de datos de cada enlace es impuesta por Linux Traffic Control (tc), que tiene varios programadores de paquetes para configurar el tráfico a una velocidad configurada. Cada host emulado tiene sus propias interfaces virtuales de Ethernet (creadas e instaladas con ip link add / set). Un par de Ethernet virtual (o veth), actúa como un cable que conecta dos interfaces virtuales o puertos de conmutador virtual; los paquetes enviados a través de una interfaz se entregan a la otra, y cada interfaz aparece como un puerto Ethernet completamente funcional para todos los sistemas y aplicaciones de software.

- Variaciones de los enlaces

Mininet permite establecer parámetros de enlace, y estos incluso pueden configurarse automáticamente desde la línea de comando (Figura 4.8):

```
$ sudo mn --link tc,bw=10,delay=10ms
```

Figura 4.8: Parámetros de enlace en Mininet

La Figura 4.9 muestra diversas formas de cómo caracterizar los enlaces entre dispositivos simulados mediante la opción "- -link", esto es muy importante, pues como veremos más adelante permite un "ajuste fino" de una topología de simulación.

a) Mininet permite ajustar valores de los enlaces, mediante el parámetro --link. Por ejemplo:

$ sudo mn --link tc, bw=100, delay=20ms

Con estos valores, se limita la velocidad de los enlaces a 100 Mbps, con un delay de 20 ms.

b) Editando la script de python que define la topología, se pueden modificar los parámetros de los distintos enlaces, por ej:

```
info( '*** Add links\n')
net.addLink(s2, s1, cls=TCLink, bw=100, delay='20ms') #enlace a 100Mbps y retardo 20 ms.
net.addLink(s2, s3, cls=TCLink, bw=100, delay='20ms', loss=2)#enlace a 100 Mbps, retardo 20 ms, y perdidas del 2%
```

addLink(<n1>, <n2>, cls=, bw=<BW>, delay=<D>,loss=<L>): Crea un enlace entre los nodos n1 y n2, con los parámetros opcionales BW(Ancho de banda), delay, y porcentaje de pérdidas L. También admite otros parámetros como jitter, latencia, etc.

```
Parameters
  bw          bandwidth in b/s (e.g. '10m')
  delay       transmit delay (e.g. '1ms' )
  jitter      jitter (e.g. '1ms')
  loss        loss (e.g. '1' )
  gro enable  GRO (False)
  txo enable  transmit checksum offload (True)
  rxo enable  receive checksum offload (True)
  speedup     experimental switch-side bw option
  use_hfsc    use HFSC scheduling
  use_tbf     use TBF scheduling
  latency_ms  TBF latency parameter
  enable_ecn  enable ECN (False)
  enable_red  enable RED (False)
  max_queue_size  queue limit parameter for
netem  Helper method: bool -> 'on'/'off'
```

c) Es posible habilitar/inhabilitar un enlace determinado:

```
mininet> link s1 h1 down
mininet> link s1 h1 up
```

Figura 4.9: Utilización del parámetro "–link" en Mininet

4.4.4. Simulación con Mininet y Controlador POX

En primer lugar, describimos el entorno de simulación, la Figura 4.10 muestra la disposición de los recursos empleados

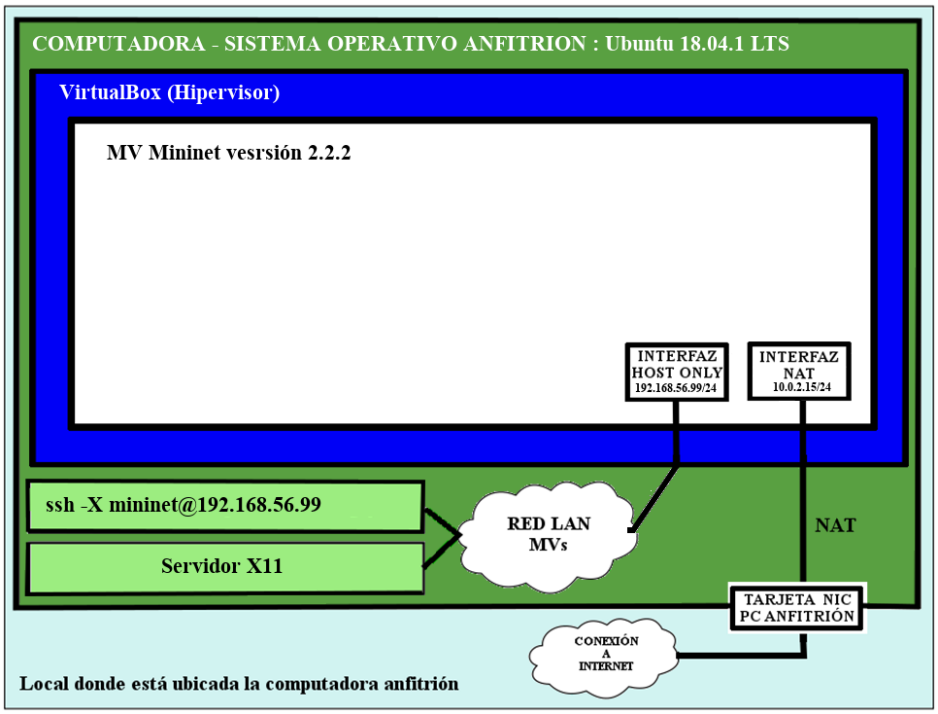

Figura 4.10: Esquema en bloques del entorno de emulación Mininet

Partiendo de este punto, la descripción se hace a manera tutorial, de forma que pueda ser reproducida sin ninguna dificultad por el lector. Abrir una sesión ssh a la MV Mininet, con el forwarding X habilitado:

- ssh -X Mininet@192.168.56.99
- Nos logueamos con user: mininet, password: mininet y obtenemos el prompt de shell: mininet@mininet-vm: $

La Figura 4.11, muestra como verificar la version de Mininet que se está utilizando, con la opción "–versión".

- Mininet@Mininet-vm:/$sudo mn –version

4.4.4.1. Arrancando la topología

Se ejecuta el comando: sudo mn –controller=remote,ip=127.0.0.1:6633 –topo tree,2,2 El comando conforma una topología de árbol, con dos niveles y dos hosts por cada switch. Se puede comprobar con el comando: net • mininet>net Obtendremos la siguiente salida: *h1 h1-eth0:s2-eth1 h2 h2-eth0:s2-eth2 h3 h3-eth0:s3-eth1 h4 h4-eth0:s3-eth2 s1 lo: s1-eth1:s2-eth3 s1-eth2:s3-eth3 s2 lo: s2-eth1:h1-eth0 s2-eth2:h2-eth0 s2-eth3:s1-eth1 s3 lo: s3-eth1:h3-eth0 s3-eth2:h4-eth0 s3-eth3:s1-eth2 c0 mininet>* El comando sudo mn –controller=remote,ip=127.0.0.1:6633 –topo tree,2,2, crea la topología de la Figura 4.12:

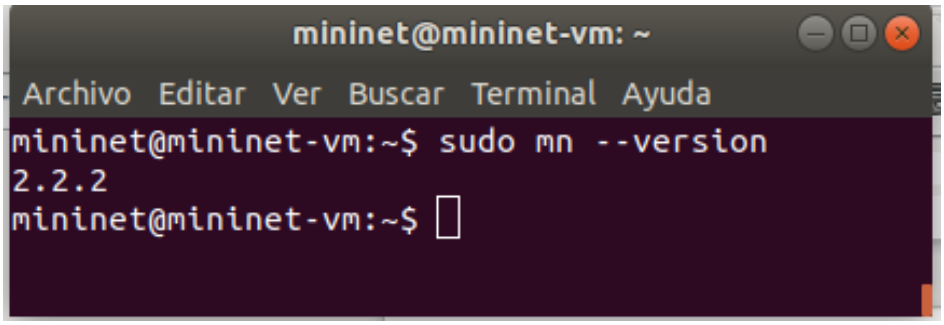

Figura 4.11: Verificando la versión de Mininet

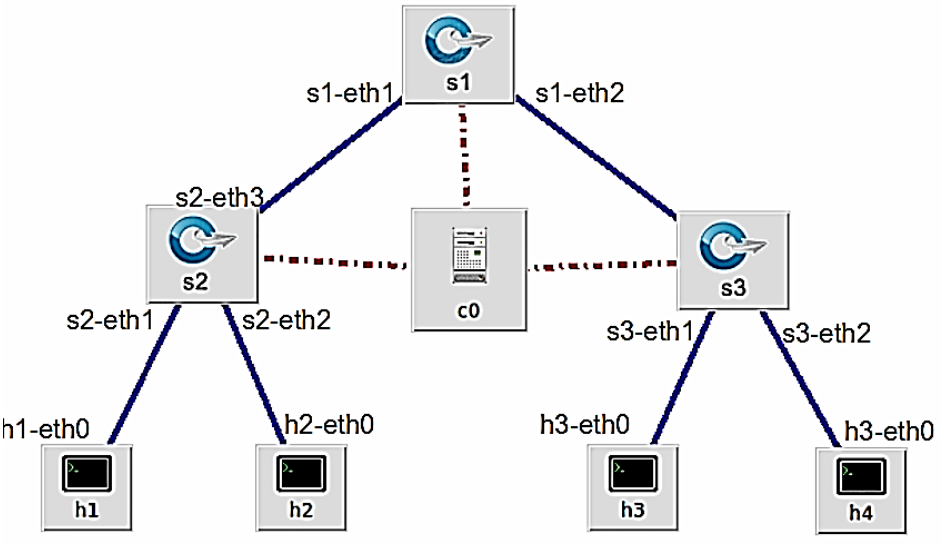

Figura 4.12

Hasta aquí se observa que tenemos tres switches OpenFlow conectados a Un controlador c0 y los switches 2 y 3 tienen a su vez conectadas dos computadoras cada uno (h1,h2 y h3,h4 respectivamente)

4.4.4.2. Arrancando el controlador

Ahora se arranca el controlador POX, para esto se inicia una nueva sesión SSH a la MV Mininet y se ejecuta el siguiente comando: Mininet@Mininet-vm: $ sudo /pox/pox.py forwarding.l2_pairs info.packet_dump samples.pretty_log log.level –DEBUG Si todo está bien el controlador se conectará con los switch de la topología y mostrará como en la Figura 4.13:

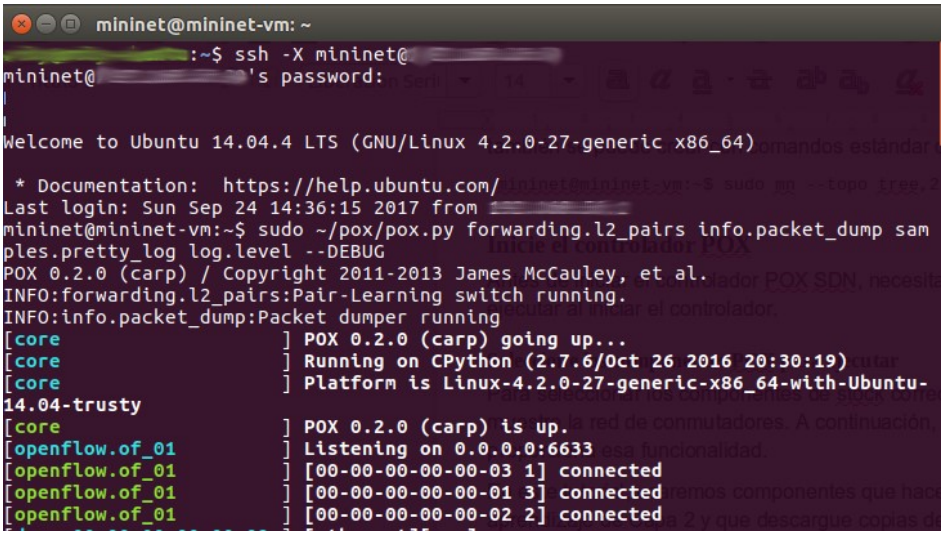

Figura 4.13: Arrancando el controlador POX

4.4.4.3. Obteniendo Información de la Red Simulada en Mininet

En primer lugar, se realizará un estudio de las funciones más básicas que ofrece Mininet, las cuales aportan información sobre los distintos nodos de la red. Y para eso se utilizan los siguientes comandos:

- nodes
- dump
- net
- ifconfig

La instrucción “nodes” enumera los componentes que forman parte de la red. En el caso del ejemplo sería:

- mininet>nodes La salida correspondiente es:

available nodes are: c0 h1 h2 h3 h4 s1 s2 s3

En el caso de la instrucción “dump”, aporta información básica de cada nodo: como la dirección IP y código PID.

- mininet>dump La salida correspondiente es: *<Host h1: h1-eth0:10.0.0.1 pid=1419><Host h2: h2-eth0:10.0.0.2 pid=1421><Host h3: h3-eth0:10.0.0.3 pid=1423><Host h4: h4-eth0:10.0.0.4*

pid=1425><OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=1430><OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None pid=1433><OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None,s3-eth3:None pid=1436><RemoteControllerip': '127.0.0.1:6633

c0: 127.0.0.1:6633 pid=1413>

El comando 'net' muestra los enlaces de la red. • mininet>net La salida correspondiente es: *h1 h1-eth0:s2-eth1 h2 h2-eth0:s2-eth2 h3 h3-eth0:s3-eth1 h4 h4-eth0:s3-eth2 s1 lo: s1-eth1:s2-eth3 s1-eth2:s3-eth3 s2 lo: s2-eth1:h1-eth0 s2-eth2:h2-eth0 s2-eth3:s1-eth1 s3 lo: s3-eth1:h3-eth0 s3-eth2:h4-eth0 s3-eth3:s1-eth2 C0* La instrucción 'ifconfig', aporta una mayor densidad de información de cada nodo. Para el caso del ejemplo, tomando el host h1: • Mininet>h1 ifconfig La salida correspondiente es: *h1-eth0 Link encap:Ethernet HWaddr 4e:95:f9:44:3d:7c inet addr:10.0.0.1 Bcast:10.255.255.255 Mask:255.0.0.0 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:0 errors:0 dropped:0 overruns:0 frame:0 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B) lo Link encap:Local Loopback inet addr:127.0.0.1 Mask:255.0.0.0 UP LOOPBACK RUNNING MTU:65536 Metric:1 RX packets:0 errors:0 dropped:0 overruns:0 frame:0 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:0 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)*

Se puede ver que el host h1, tiene la interface h1-eth0, además del loopback (lo). Se tiene que tener en cuenta que h1-eth0 no es "vista" por el sistema principal Linux al ejecutar el comando ifconfig, debido a que son específicas para el espacio de nombres de la red del proceso host. Otro dato a tener en cuenta es que debido a que estas interfaces representan parte de un par ethernet virtual, el host no puede hacerse un ping a si mismo desde la interfaz. Sin embargo, en el caso de ejecutar ifconfig en el switch s1 se pueden observar las distintas interfaces, junto con la conexión de salida de la máquina virtual (eth0).

eth1 Link encap:Ethernet HWaddr 08:00:27:f3:ed:f0 inet addr:192.168.56.99 Bcast:192.168.56.255 Mask:255.255.255.0 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:303 errors:0 dropped:0 overruns:0 frame:0 TX packets:217 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:29666 (29.6 KB) TX bytes:35222 (35.2 KB)

lo Link encap:Local Loopback inet addr:127.0.0.1 Mask:255.0.0.0 UP LOOPBACK RUNNING MTU:65536 Metric:1 RX packets:4826 errors:0 dropped:0 overruns:0 frame:0 TX packets:4826 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:0 RX bytes:253972 (253.9 KB) TX bytes:253972 (253.9 KB)

s1 Link encap:Ethernet HWaddr b6:12:c2:ca:d8:40 UP BROADCAST RUNNING MTU:1500 Metric:1 RX packets:0 errors:0 dropped:0 overruns:0 frame:0 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:0 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

s2 Link encap:Ethernet HWaddr ba:cd:2f:7d:51:43 UP BROADCAST RUNNING MTU:1500 Metric:1 RX packets:0 errors:0 dropped:0 overruns:0 frame:0 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:0 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

s3 Link encap:Ethernet HWaddr 16:84:49:92:be:4f UP BROADCAST RUNNING MTU:1500 Metric:1 RX packets:0 errors:0 dropped:0 overruns:0 frame:0 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:0 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

s1-eth1 Link encap:Ethernet HWaddr 0e:29:21:27:13:50 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:0 errors:0 dropped:0 overruns:0 frame:0 TX packets:0 errors:0 dropped:0

overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

s1-eth2 Link encap:Ethernet HWaddr b6:c3:2f:99:d3:3e UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:0 errors:0 dropped:0 overruns:0 frame:0 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

s2-eth1 Link encap:Ethernet HWaddr 8a:e4:c9:cf:7f:57 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:0 errors:0 dropped:0 overruns:0 frame:0 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

s2-eth2 Link encap:Ethernet HWaddr 2e:f6:49:11:f3:e8 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:0 errors:0 dropped:0 overruns:0 frame:0 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

s2-eth3 Link encap:Ethernet HWaddr 0e:f8:d0:4d:67:e6 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:0 errors:0 dropped:0 overruns:0 frame:0 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

s3-eth1 Link encap:Ethernet HWaddr a2:58:0d:00:a9:a4 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:0 errors:0 dropped:0 overruns:0 frame:0 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

s3-eth2 Link encap:Ethernet HWaddr ee:a6:b9:88:b2:43 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:0 errors:0 dropped:0 overruns:0 frame:0 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

s3-eth3 Link encap:Ethernet HWaddr 46:da:3a:77:95:4f UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:0 errors:0 dropped:0 overruns:0 frame:0 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B) *Mininet>s1 ifconfig eth0 Link encap:Ethernet HWaddr 08:00:27:c4:18:cb inet addr:10.0.2.15 Bcast:10.0.2.255 Mask:255.255.255.0 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:393 errors:0 dropped:0 overruns:0 frame:0 TX packets:398 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:36702 (36.7 KB) TX bytes:34902 (34.9 KB)*

eth1 Link encap:Ethernet HWaddr 08:00:27:f3:ed:f0 inet addr:192.168.56.99 Bcast:192.168.56.255 Mask:255.255.255.0 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:303 errors:0 dropped:0 overruns:0 frame:0 TX packets:217 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:29666 (29.6 KB) TX bytes:35222 (35.2 KB)

lo Link encap:Local Loopback inet addr:127.0.0.1 Mask:255.0.0.0 UP LOOPBACK RUNNING MTU:65536 Metric:1 RX packets:4826 errors:0 dropped:0 overruns:0 frame:0 TX packets:4826 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:0 RX bytes:253972 (253.9 KB) TX bytes:253972 (253.9 KB)

s1 Link encap:Ethernet HWaddr b6:12:c2:ca:d8:40 UP BROADCAST RUNNING MTU:1500 Metric:1 RX packets:0 errors:0 dropped:0 overruns:0 frame:0 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:0 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

s2 Link encap:Ethernet HWaddr ba:cd:2f:7d:51:43 UP BROADCAST RUNNING MTU:1500 Metric:1 RX packets:0 errors:0 dropped:0 overruns:0 frame:0 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:0 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

s3 Link encap:Ethernet HWaddr 16:84:49:92:be:4f UP BROADCAST RUNNING MTU:1500 Metric:1 RX packets:0 errors:0 dropped:0 overruns:0 frame:0 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:0 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

s1-eth1 Link encap:Ethernet HWaddr 0e:29:21:27:13:50 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:0 errors:0 dropped:0 overruns:0 frame:0 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

s1-eth2 Link encap:Ethernet HWaddr b6:c3:2f:99:d3:3e UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:0 errors:0 dropped:0 overruns:0 frame:0 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

s2-eth1 Link encap:Ethernet HWaddr 8a:e4:c9:cf:7f:57 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:0 errors:0 dropped:0 overruns:0 frame:0 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

s2-eth2 Link encap:Ethernet HWaddr 2e:f6:49:11:f3:e8 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:0 errors:0 dropped:0 overruns:0 frame:0 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

s2-eth3 Link encap:Ethernet HWaddr 0e:f8:d0:4d:67:e6 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:0 errors:0 dropped:0 overruns:0 frame:0 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

s3-eth1 Link encap:Ethernet HWaddr a2:58:0d:00:a9:a4 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:0 errors:0 dropped:0 overruns:0 frame:0 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

s3-eth2 Link encap:Ethernet HWaddr ee:a6:b9:88:b2:43 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:0 errors:0 dropped:0 overruns:0 frame:0 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

s3-eth3 Link encap:Ethernet HWaddr 46:da:3a:77:95:4f UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:0 errors:0 dropped:0 overruns:0 frame:0 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B) Esto es debido a que el switch s1 corre en el espacio de root en el kernel del sistema anfitrión, por ende, muestra todas las interfaces (incluyendo las del anfitrión MV Mininet).

4.4.5. Comprobando la conectividad en la topología

4.4.5.1. ping

La sintaxis de la instrucción que hay que introducir dentro del CLI de Mininet es la siguiente: host Origen ping -c [número de Paquetes] host Destino Ejemplo: mininet>h1 ping -c 3 h3 PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data. 64 bytes from 10.0.0.3: $icmp_seq = 1 ttl = 64 time = 109 ms 64 bytes from 10,0,0,3 : icmp_seq = 2 ttl = 64 time = 0,402 ms 64 bytes from 10,0,0,3 : icmp_seq = 3 ttl = 64 time = 0,129 ms --- 10,0,0,3 ping statistics --- 3 packets transmitted, 3 received, 0 rtt min/avg/max/mdev = 0,129/36,639/109,387/51,440 ms$

Como se puede observar el host 1 (h1) hace un ping al host 3 (h3), enviándole 3 paquetes, además el sistema muestra las estadísticas del envío por pantalla (número de paquetes enviados, número de paquetes recibidos, porcentaje de paquetes perdidos y los tiempos de envío total, mínimo, promedio y máximo). Para analizar más en detalle el intercambio de las tramas entre ambos hosts en la red, volveremos sobre esto en el punto 4.5.4 Se puede analizar el tráfico de control de OpenFlow: El primer host realiza un ARP para conocer la dirección MAC del segundo, esto causa un mensaje "$packet_in$*haciaelcontrolador.Elcontroladorposteriormenterealizaunbroadcasthaciaelrestodelospuertosdelared(enesteca*

4.4.5.2. pingall

*pingall realiza un ping entre todos los hosts de la red y sirve para comprobar en forma completa la conectividad entre todos los hosts. mininet>pingall *** Ping: testing ping reachability h1 ->h2 h3 h4 h2 ->h1 h3 h4 h3 ->h1 h2 h4 h4 ->h1 h2 h3 *** Results: 0 % dropped (12/12 received)*

4.4.5.3. pingpair

*pingpair es muy similar a pingall, la cual realiza un ping entre las distintas parejas de hosts de la red. mininet>pingpair h1 ->h2 h2 ->h1 *** Results: 0 % dropped (2/2 received)*

4.4.5.4. iperf

*Se trata de un test o prueba que se ejecuta en un servidor iperf residente en un host y un cliente iperf residente en un segundo host, para caracterizar el ancho de banda entre ambos. (si el comando se escribe sin argumentos, normalmente hace la prueba entre el primero y el último host, que en nuestro ejemplo serían h1 y h4) mininet>iperf *** Iperf: testing TCP bandwidth between h1 and h4 *** Results: ['47.0 Gbits/sec', '47.1 Gbits/sec']*

Mas adelante volveremos sobre iperf (versiones 2 y 3), que incorporan interesantes funcionalidades y permiten mayor control sobre la simulación.

4.5. Estudio del protocolo OpenFlow con Mininet

Para efectuar las primeras pruebas con una topología y poder analizar el intercambio de mensajes del protocolo OpenFlow, aprovechamos lo aprendido en los puntos anteriores y establecemos un escenario de prueba como el que muestra el siguiente punto.

4.5.1. Estableciendo el entorno para la simulación y las pruebas

A continuación, se configura el escenario para una simulación que permite ver en acción al protocolo OpenFlow, y el intercambio de mensajes analizados con wireshark. En orden a ello, se sugiere seguir puntualmente las siguientes instrucciones: • Arrancamos cuatro terminales en la MV Mininet: Se procede a abrir cuatro veces la terminal de Ubuntu, teniendo cuidado que las mismas se abran en pestañas, (Figura 4.14). Esto se organiza así, para poder trabajar simultáneamente con distintas instrucciones y permitir observar los resultados en forma cómoda y ordenada. • En la Terminal 1:

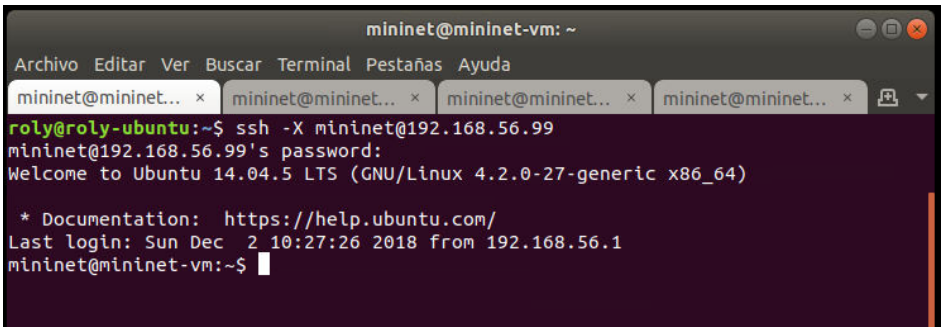

Figura 4.14: Disposición de las terminales para iniciar la simulación

Arrancamos wireshark en la interfaz lo de la MV Mininet (Figura 4.15):

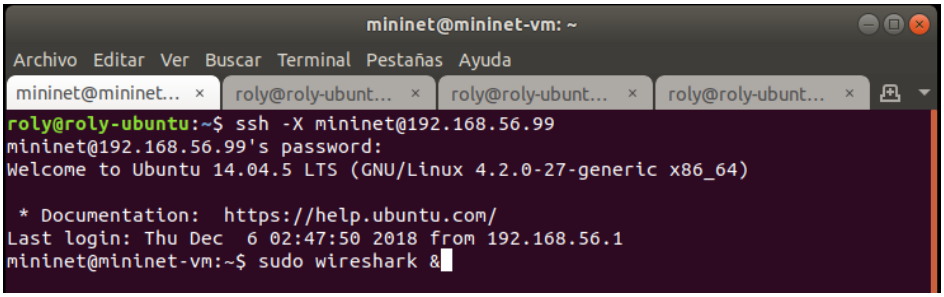

Figura 4.15: Arrancando wireshark

Se debe configurar el wireshark para que capture tráfico a través de la interfaz loopback de la MVmininet, y aplicar el filtro openflow$_v$1, queeslaversióndelprotocoloconelquetrabajaelcontroladorPOXquevieneincorporadoenla

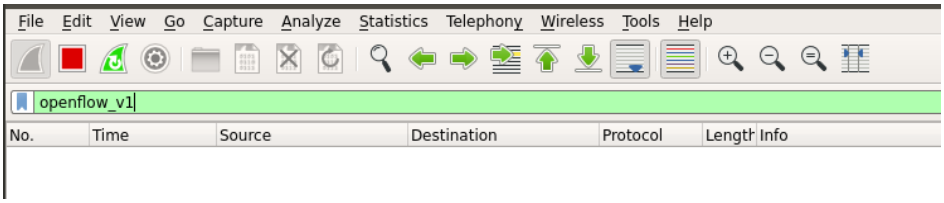

Figura 4.16: Wireshaark configurado para capturar en la interfaz loopback y con filtro openflow$_v$1

• *En la Terminal 2: Arrancamos el controlador POX (se encuentra incluido en la MV Mininet, así que solo se escribe el comando sudo /pox/pox.py forwarding.l2_pairs info.packet_dump samples.pretty_log log.level –DEBUG; como muestra la Figura 4.17)*

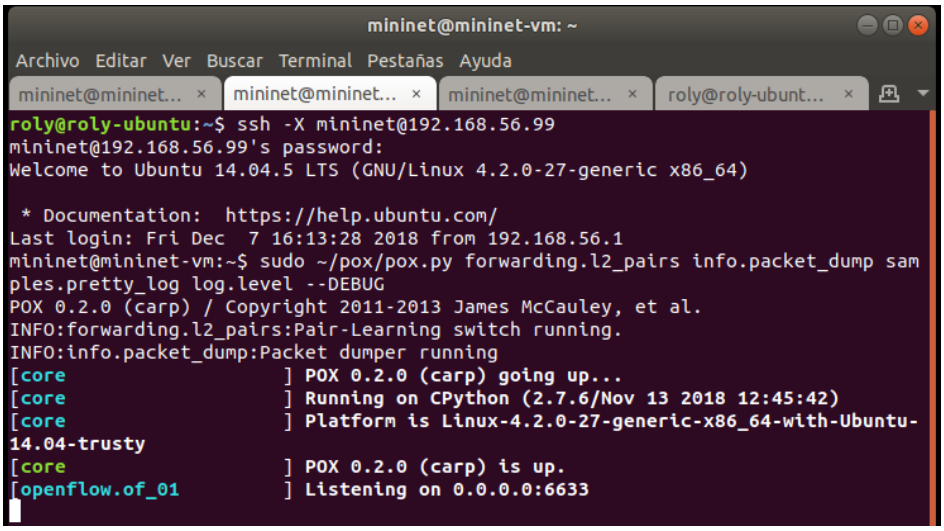

Figura 4.17: Arrancando el controlador POX

• *Como se puede ver en la Figura 4 17, el controlador implementa OpenFlow versión 1 y escucha en el puerto 6633 y emula un switch de autoaprendizaje forwarding.l2_pairs (aprende y genera las reglas para la tabla de flujos, ver [69]. • En la Terminal3: Arrancamos la topología : sudo mn –controller=remote,ip=127.0.0.1:6633 –topo=tree,2,2 –mac (Figura 4.18):*

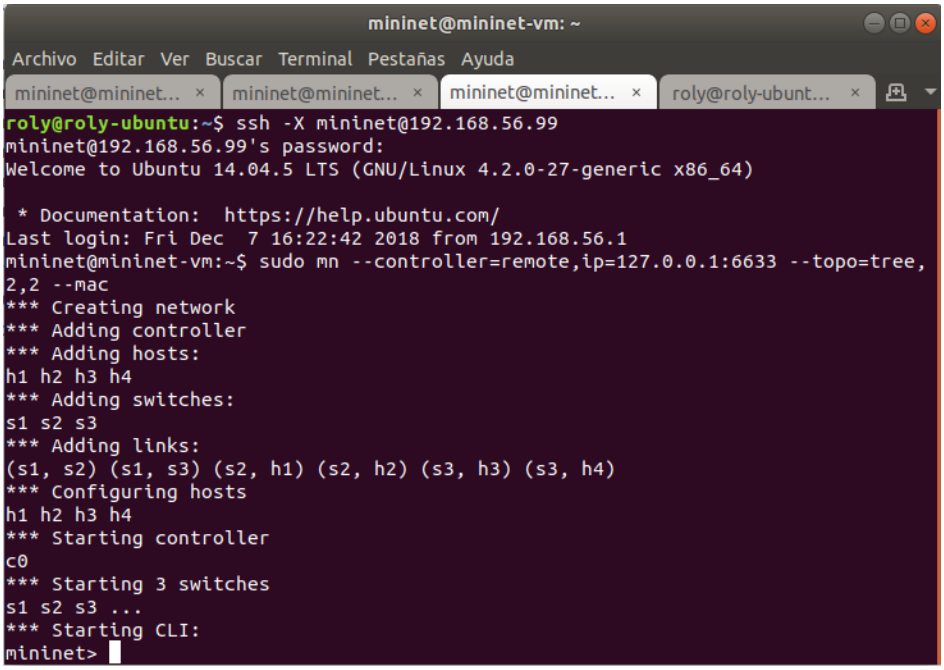

Figura 4.18: Arrancando la topología de prueba

• *En la Terminal 4: La utilizaremos para comandos ovs (Open vSwitch) desde afuera del cli Mininet*

De esta manera queda conformado el escenario de simulación como muestra la Figura 4.19. Hasta

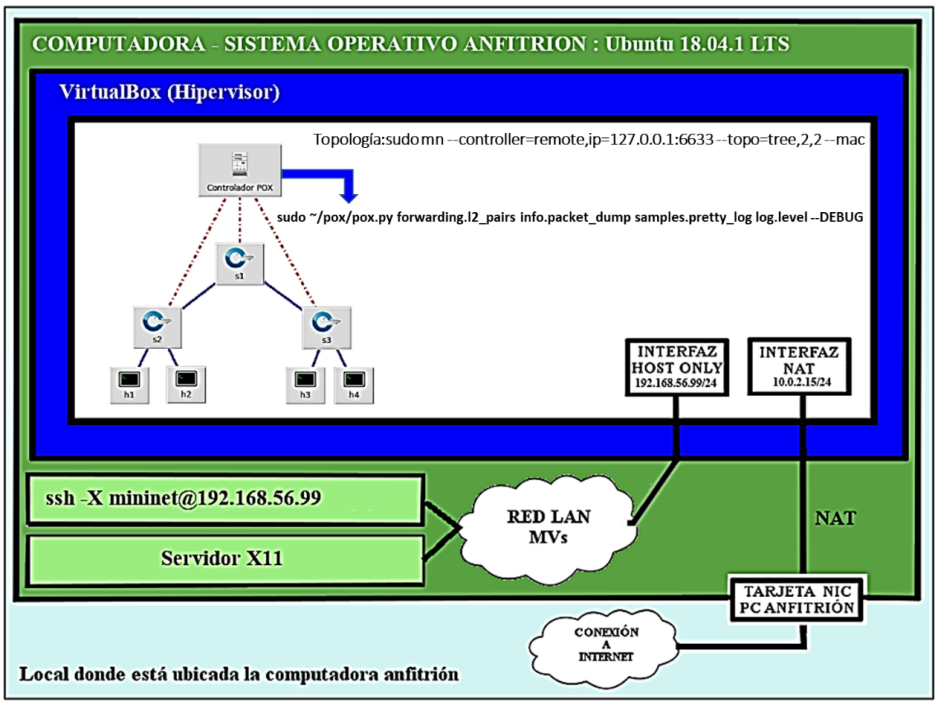

Figura 4.19: Entorno de simulación con una topología de árbol y cuatro hosts para pruebas con OpenFlow

aquí tenemos preparado todo lo necesario para la simulación y poder comenzar, por ejemplo, viendo

el intercambio de mensajes OpenFlow. Sin embargo, antes de proseguir, consideramos importante remarcar ciertos aspectos de la implementación de los dispositivos puntos (), con relación a algunos detalles sobre el controlador POX, y a los comandos "ovs" utilizados y en estas primeras experiencias.

4.5.1.1. Detalles del controlador POX

POX se invoca ejecutando pox.py o debug-pox.py. El primero es para correr en circunstancias normales. Este último está pensado para cuando estás tratando de depurar problemas (es una buena idea usar debug-pox.py al realizar el desarrollo). El propio POX tiene argumentos opcionales de línea de comando que pueden usarse al inicio de la línea de comando. En nuestro ejemplo:

sudo /pox/pox.py forwarding.l2$_p$airsinfo.packet$_d$umpsamples.pretty$_l$oglog.level˜DEBUG

Cuando hablamos de componentes en POX, lo que realmente queremos decir es algo que podemos poner en la línea de comando de POX. En los siguientes párrafos, se describen algunos de los componentes que vienen con POX. Se muestran a manera de ejemplo, solamente tres componentes, pero son muchos y conviene leer la documentación que está en el sitio [92], a los efectos de experimentar con distintos tipos de switches y sus respectivos comportamientos, el sitio también tiene excelentes tutoriales para experimentar con POX. • Py: Este componente hace que POX inicie un intérprete de Python interactivo que puede ser útil para la depuración y la experimentación interactiva. Otros componentes pueden agregar funciones / valores al espacio de nombres de este intérprete. • forwarding.l2_learning: Este componente hace que los conmutadores OpenFlow actúen como un tipo de switch de aprendizaje L2. Si bien este componente aprende las direcciones L2, los flujos que instala son coincidencias exactas en tantos campos como sea posible. Por ejemplo, diferentes conexiones TCP darán lugar a que se instalen diferentes flujos. • forwarding.l2_pairs: Como l2_learning, este componente también hace que los conmutadores OpenFlow actúen como un tipo de switch de aprendizaje L2. Sin embargo, esta es probablemente la forma más sencilla de hacerlo correctamente. A diferencia de l2_learning, l2_pairs instala reglas basadas únicamente en direcciones MAC.

4.5.1.2. Sobre las herramientas de comando en línea ovs-dpctl, ovs-ofctl y ovs-vsctl

Durante el desarrollo del trabajo nos encontramos con algunas particularidades al utilizar ovs-dpctl [93], ovs-ofctl [94] y ovs-vsctl [95]. Estas herramientas son muy utilizadas para el monitoreo y administración de switches OpenFlow. La aparente similitud entre ambas herramientas a veces confunde; por ese motivo elaboramos la Figura 4.20 que muestra claramente el ámbito de aplicación de cada una de ellas.

Como muestra la Figura 4.20, podemos ejecutar los comandos ovs-dpctl, ovs-ofctl y ovs-vsctl desde el promt de la una terminal en la MVMininet, y también podemos utilizar el comando dpctl desde el cli de Mininet, en el desarrollo del tutorial veremos en detalle la aplicación de estos comandos.

4.5.2. Estableciendo la comunicación Controlador - Switch con OpenFlow

A los efectos de ir avanzando en forma ordenada y secuencial en el estudio de los mensajes, en los siguientes puntos, se describen las distintas operaciones a realizar, conforme a lo que, en nuestra experiencia, resulto ser la manera más conveniente desde el punto de vista didáctico. También aclaramos que existen muchos y excelentes tutoriales y documentos en relación con esto, pero decidimos explicar los más detalladamente dentro de nuestras posibilidades, y ¿por qué no?, comentar los inconvenientes

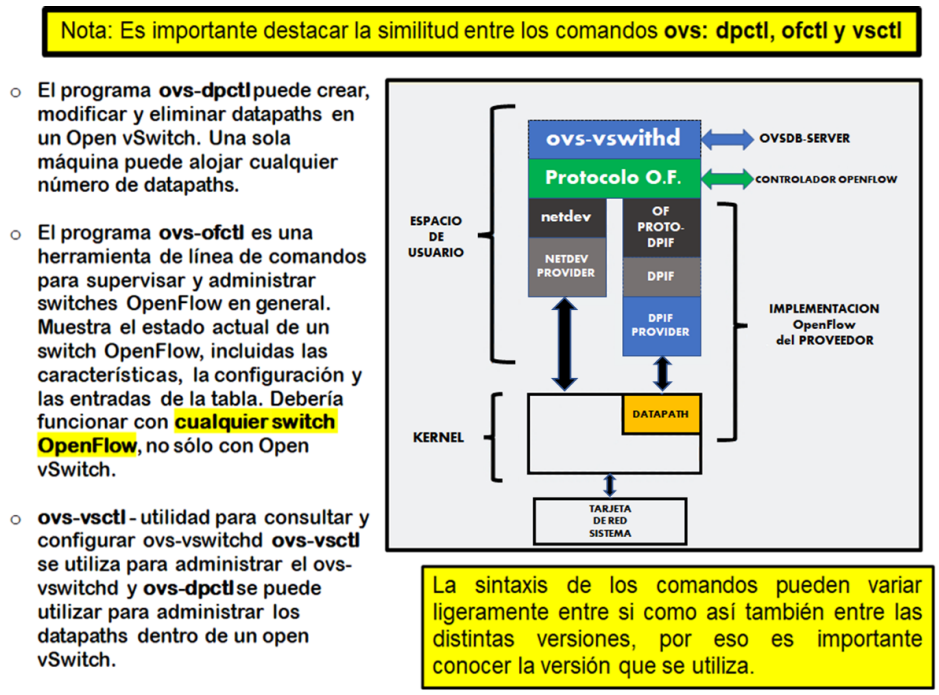

Figura 4.20: Comandos ovs-(dpctl , ofctl y vsctl)

que tuvimos (equivocaciones), exponiendo algunos tips, para que sirvan de advertencia a quienes encaran un estudio inicial y ganen tiempo aprendiendo de nuestros "errores" . Bien, dicho todo esto, sin más, ¡avancemos con los mensajes!

4.5.2.1. Mensaje OFPT_HELLO

Cuando se establece inicialmente una conexión OpenFlow (2.10.1) sucede los siguiente:

- *1º) Cada lado de la conexión debe enviar inmediatamente un mensaje de OFPT_HELLO, que contendrá el campo versión con el valor de está, se asume que la conexión se establece para dispositivos que soporten solamente la versión 1, ya que es la versión de la especificación del protocolo OpenFlow que se está trabajando (implementación de POX).*
- *2º) Tras la recepción de este mensaje, el receptor calculará la versión del protocolo OpenFlow a ser usado entre el número de versión que envió y la recibida.*
- *3º) Si la versión de protocolo negociada es soportada, la conexión procede. De lo contrario, el receptor debe responder con un mensaje OFPT_ERROR con un Tipo de campo de OFPET_HELLO_FAILED, un campo de código de OFPHFC_COMPATIBLE, y opcionalmente una cadena ASCII explicando la situación en los datos, y luego terminar la conexión.*

Tip: *Previamente a lanzar la topología con Mininet, no olvidar arrancar la captura con wireshark, con el filtro openflow_v1 (Figura 4.22), de esta manera, desde el inicio, se puede observar el valor de los campos más relevantes de la secuencia descrita en 4.5.2.1.-*

En la terminal 3, lanzamos la topología como se indica en la Figura 4.18, luego observamos la terminal 2 donde se encuentra corriendo el controlador POX y observamos que se produjo la conexión entre los switches y el controlador (Figura 4.21). 4.21

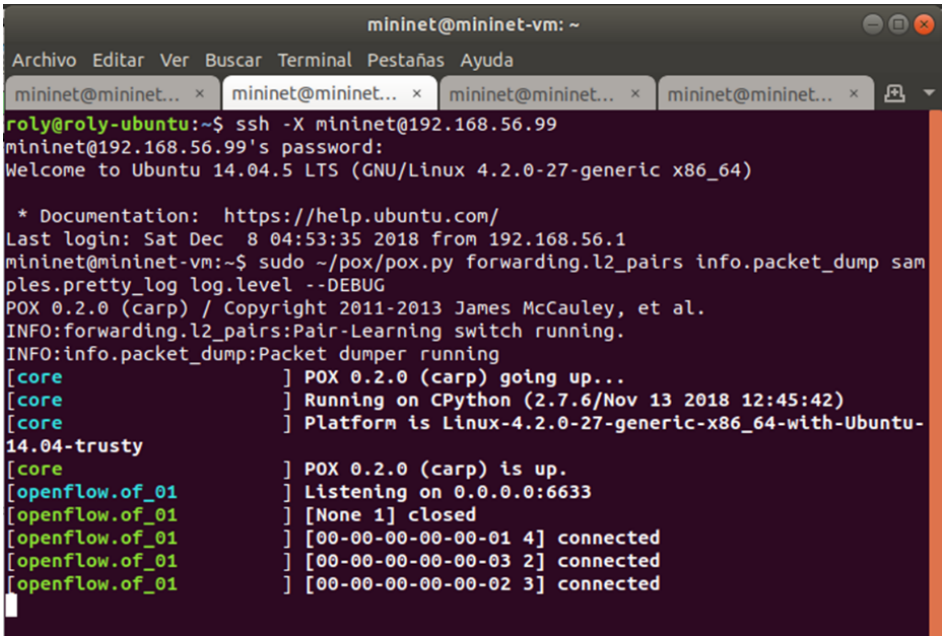

Figura 4.21: Controlador POX conectado

Observemos el intercambio en detalle, viendo como la terminal del controlador POX (Figura 4.21), muestra que se conectaron los switches,

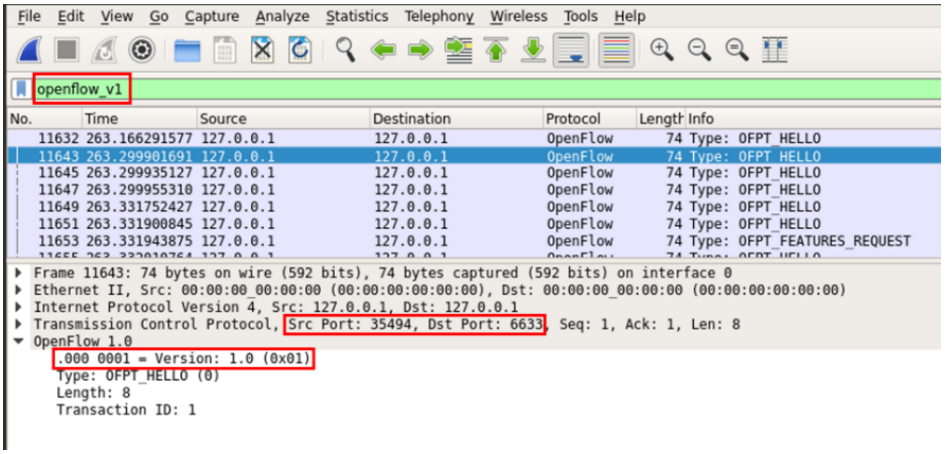

Figura 4.22: Intercambio inicial mensajes OFPT_HELLO

La Figura 4.22, muestra un mensaje OFPT_HELLO, de un switch hacia el controlador (Src Port: 35494, Dst Port: 6633), indicando que trabaja con protocolo OpenFlow Versión: 1.0 (0x01). Luego el controlador envía OFPT_HELLO hacia el switch (Figura 4.23)

Como se puede observar en Figura 4.22 y Figura 4.23, ambos coinciden y acuerdan en utilizar el protocolo OpenFlow versión 1.0. Ahora veamos que sucede si un switch que implementa OpenFlow versión 1.3 intenta negociar con un controlador que solo implementa OpenFlow versión 1.0 (precisamente el caso de POX). Se pone en marcha la misma topología, pero esta vez con switches implementando OpenFlow versión 1.3 de la siguiente manera: $ sudo mn –controller=remote,ip=127.0.0.1:6633 –topo=tree,2,2 –switch=ovsk, protocols=OpenFlow13 –mac La parte resaltada en amarillo, indica de

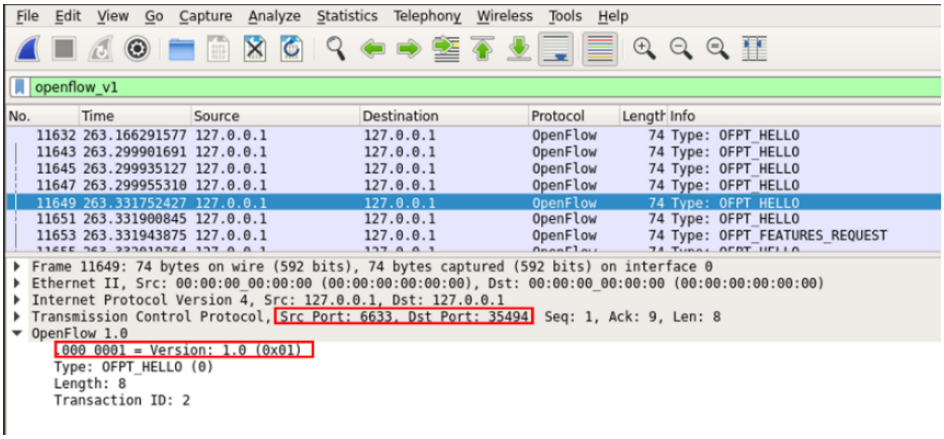

Figura 4.23: Mensaje OFPT_HELLO de controlador a switch

que tipo son los switches de la topología (OpenFlow versión 1.3 en este caso). La Figura 4.24, muestra el mensaje OFPT_HELLO dirigido al controlador (POX en nuestro caso), prestar atención al campo versión, y además que tipo de filtro se aplica a la captura, pues en este caso vamos a tener mensajes con dos versiones distintas de OpenFlow, la 1.0 y la 1.3

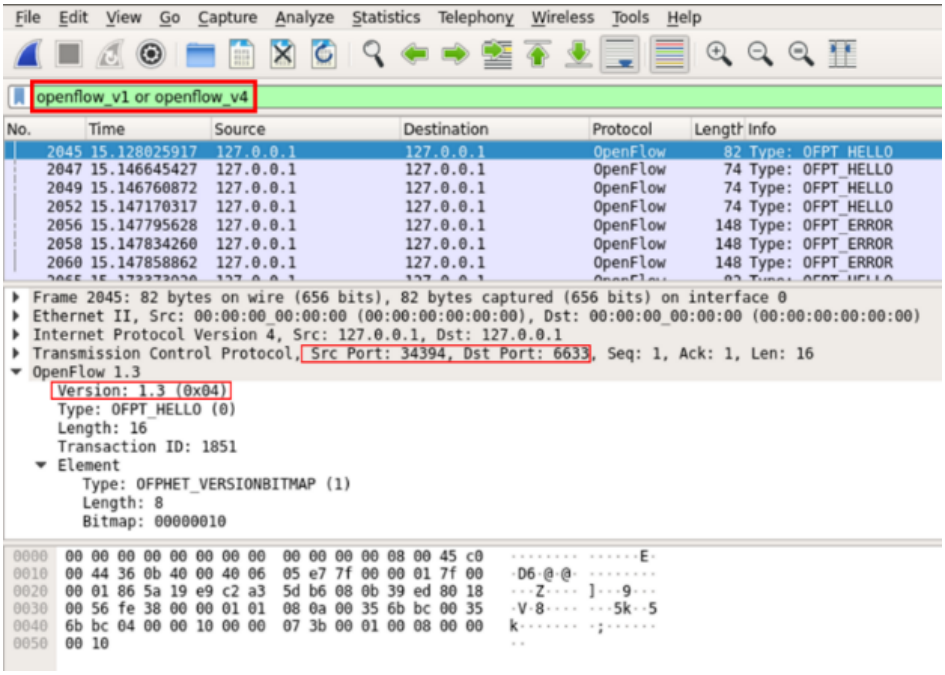

Figura 4.24: Mensaje OFPT_HELLO de un switch corriendo OpenFlow 1.3

La Figura 4.25 muestra el mensaje OFPT_HELLO, desde al controlador al switch.

Obviamente los campos de versión en ambos casos son distintos, esto corresponde a la tercera situación descrita en 4.5.2.1. Por lo tanto, se producirá el mensaje de OFPT_ERROR con una porción en código ASCII explicando la situación en los datos, y luego terminar la conexión (Figura 4.26).

Por último y para completar se muestra en la Figura 4.27 que sucede ante esta situación en la terminal del controlador POX, donde se observa el informe de error y luego la terminación de la

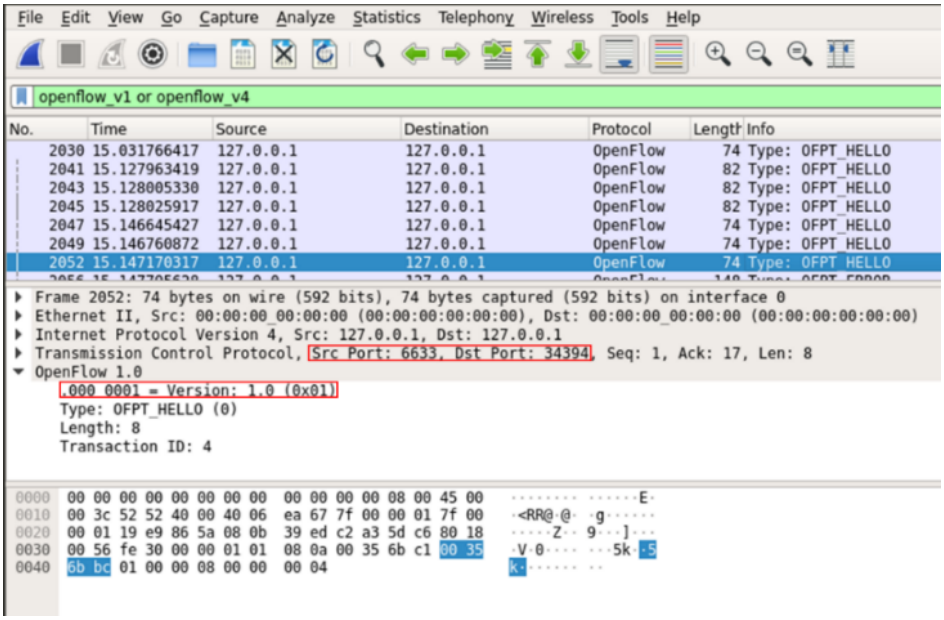

Figura 4.25: Mensaje OFPT_HELLO de Controlador con OpenFlow versión 1.0 hacia el switch con OpenFlow versión 1.3

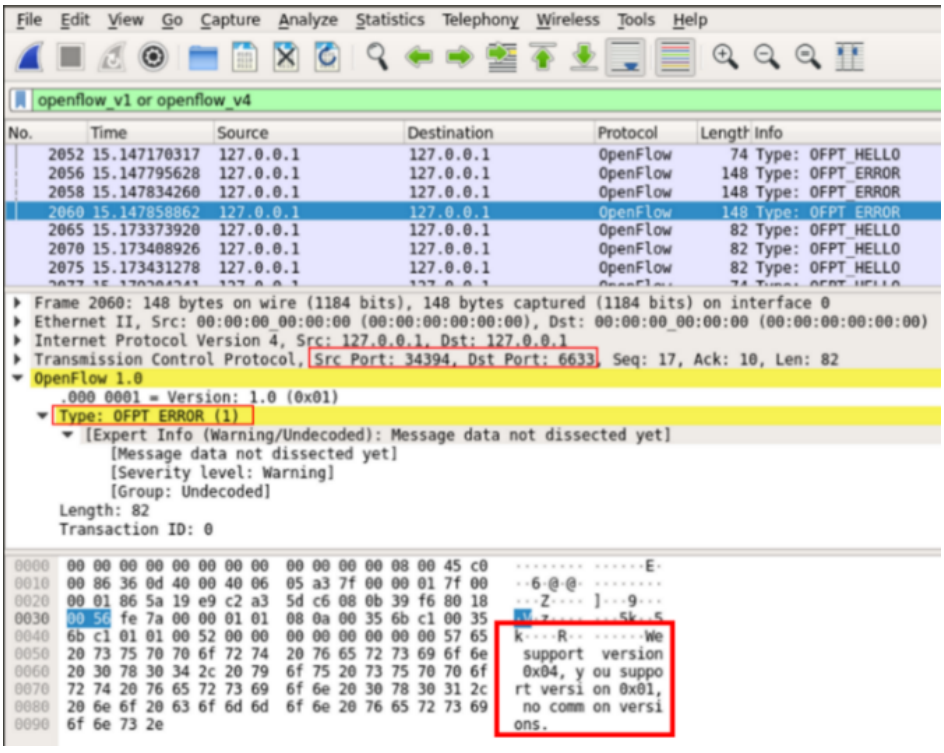

Figura 4.26: Mensaje OFPT_ERROR por versión distinta de OpenFlow

conexión (ver 2.8)

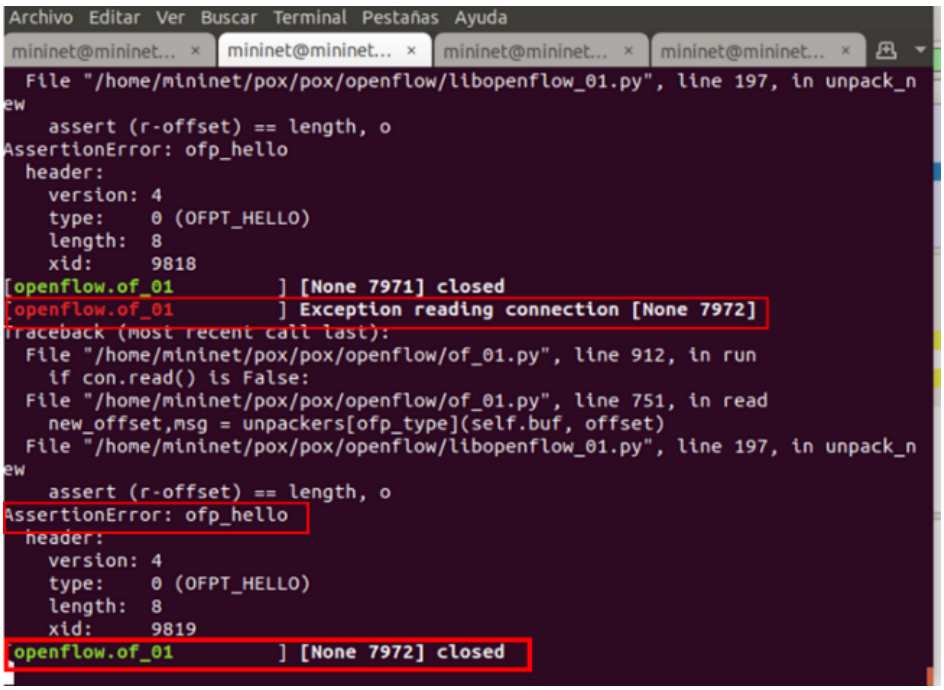

Figura 4.27: Controlador POX cerrando la conexión por error en diferencia de versión de OpenFlow

4.5.2.2. Mensaje OFPT_FEATURES_REQUEST

Tip: *Para una mayor claridad en la pantalla, Recomendamos que en la captura con wireshark, se active primero el filtro openflow_v1, luego se utilice la herramienta de análisis (Analyze - Follow - TCP Stream) sobre un mensaje Hello, de uno cualquiera de los switch, y de este modo conseguimos ver todo el intercambio entre el switch escogido y el controlador (conversación switch/controlador Figura 4.28). Obviamente los diálogos con los switches restantes serán idénticos en lo que hace al mecanismo de intercambio de mensajes de características y configuración que es lo que nos interesa en este punto.*

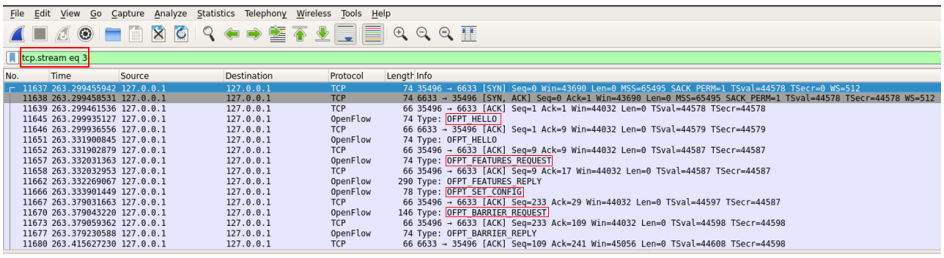

Figura 4.28: Dialogo entre un switch y el controlador

Como se observa en la secuencia de mensajes de la Figura 4 28, el controlador solicita las características del dispositivo, dirigiéndole un mensaje OFPT_FEATURES_REQUEST,Figura 4.29

El dispositivo, contesta la solicitud mediante un mensaje OFPT_FEATURES_REPLAY, Figura 4.30, Como se observa en la tanto el mensaje de solicitud, como el de respuesta de las características del dispositivo, están dirigido (y respondido) por el switch s2 o datapath 2 (mostrado en hexadecimal

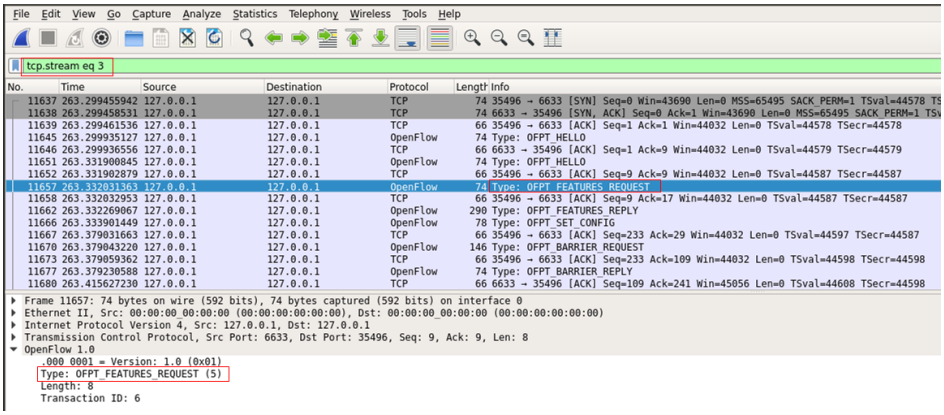

Figura 4.29: Mensaje OFPT_FEATURES_REQUEST

en el campo Datapath unique ID con un recuadro rojo). Otro detalle importante en este mensaje, es la

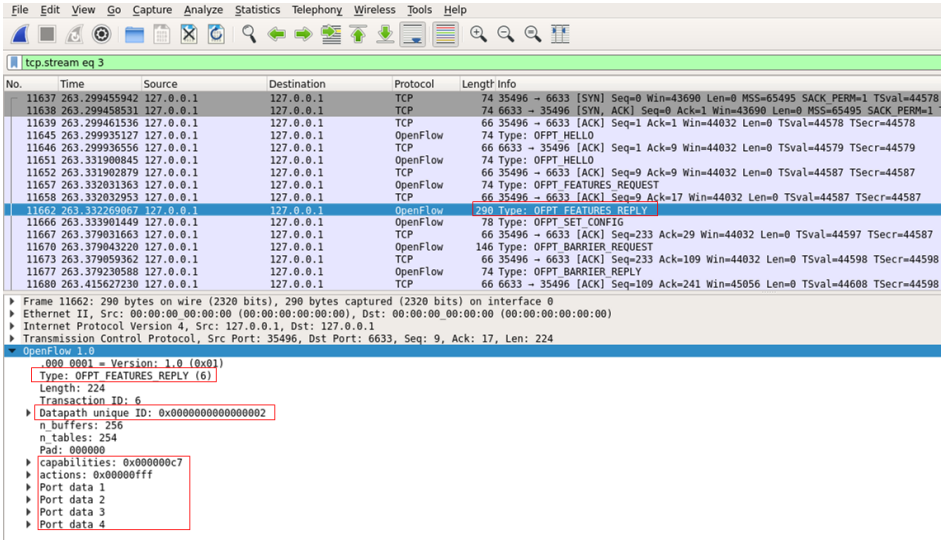

Figura 4.30: mensaje OFPT_FEATURES_REPLAY

descripción (mostrada en el recuadro de la parte inferior) de:

- *Características del switch (campo capabilities)*
- *Acciones implementadas (campo actions)*
- *Características de los puertos (campo port data 1,2...)*

El detalle de los campos mencionados en el párrafo anterior es mostrado en los siguientes puntos, Figura 4.31, y Figura 4.32.

Detalle de los campos capabilities y action del switch o datapath

Los bits puestos en "1", que muestra la Figura 4 31, están indicando las capacidades del switch s2, y en el caso del campo actions, indican que acciones pueden ser invocadas como argumentos en las entradas de la tabla de flujos

Detalle del campo Port data 1 *Esta figura muestra el detalle del campo Port data 1, del mensaje OFPT_FEATURES_REPLAY, note que los bits puestos en "1", indican las características presentes*

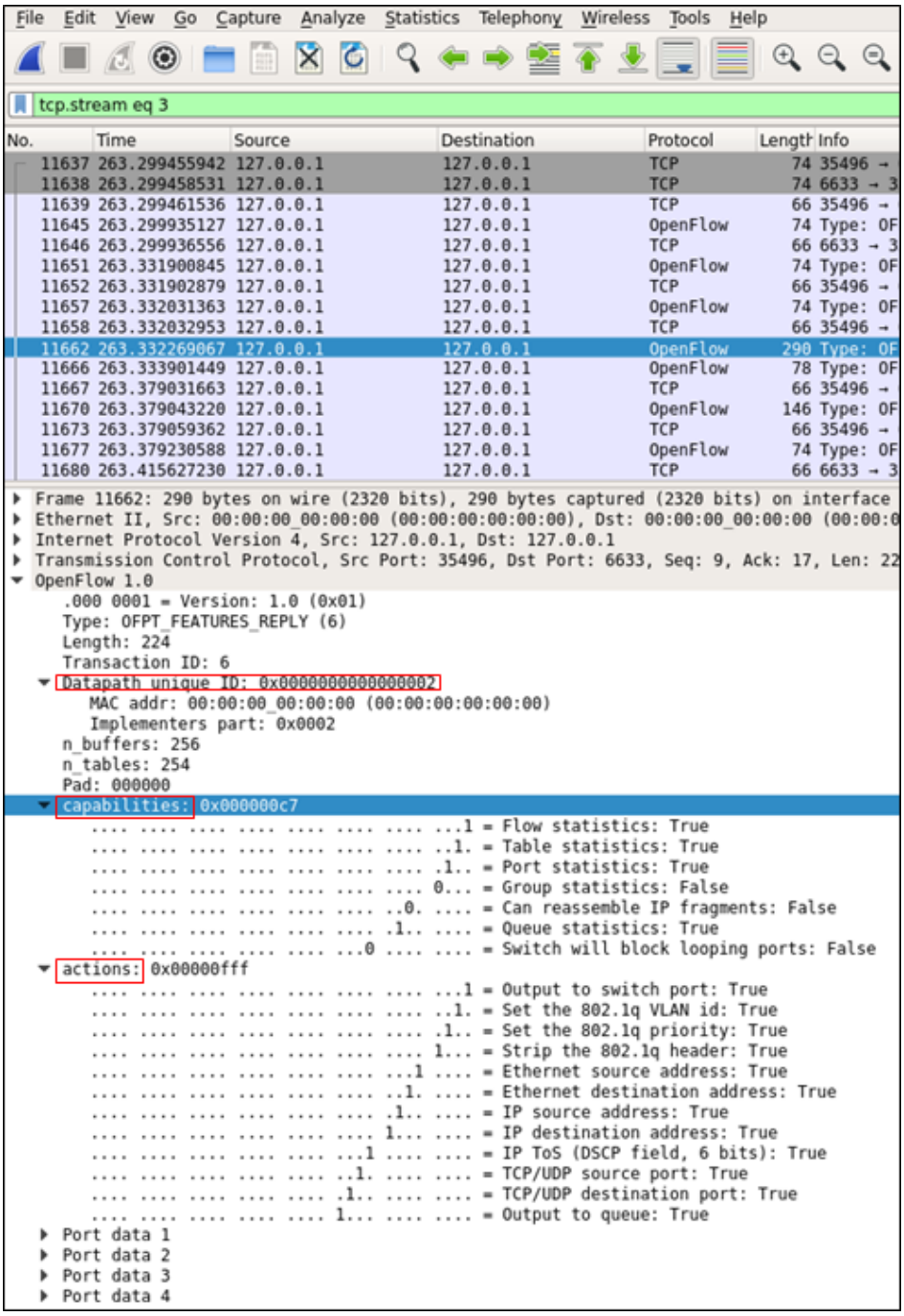

Figura 4.31: Descripción de los campos capabilities y action del switch s2

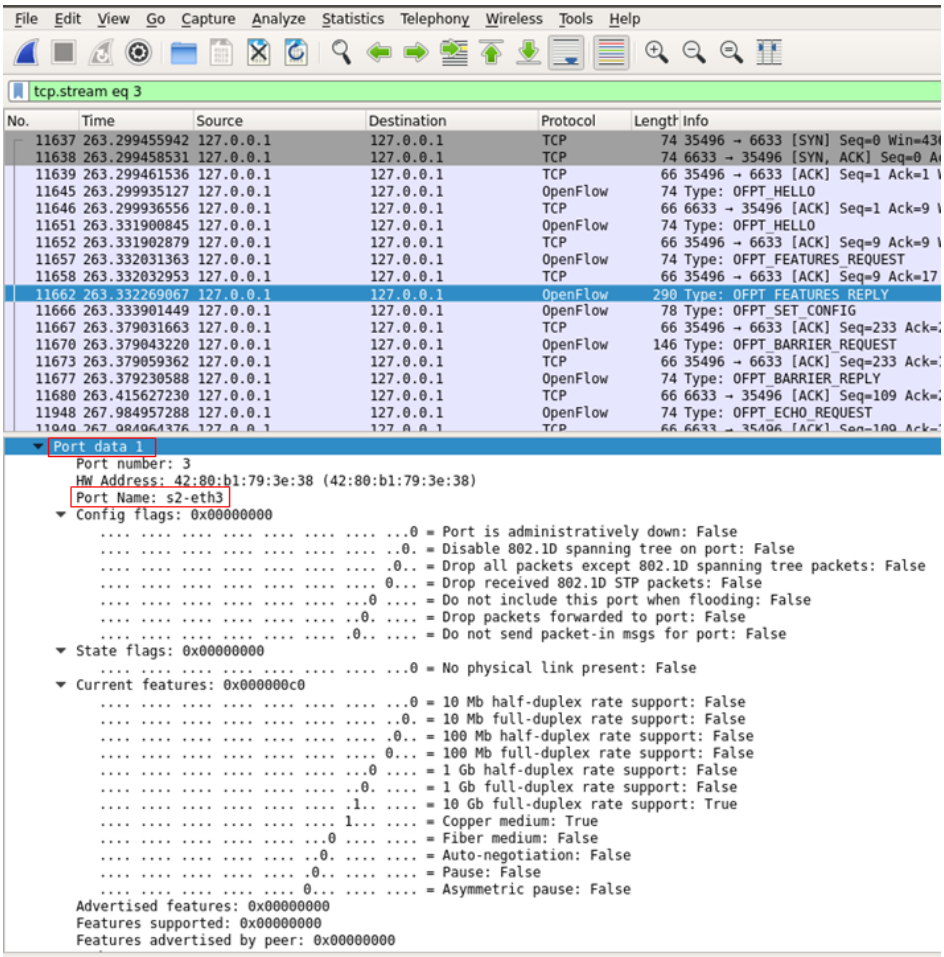

Figura 4.32: Detalle del campo Port data 1

en el puerto mencionado, en este ejemplo es el puerto "3", del switch s2. Indica que se trata de un puerto a 10Gb. full-duplex, para cable de cobre.

4.5.2.3. Mensaje OFPT_SET_CONFIG

El controlador puede establecer y consultar los parámetros de configuración en el switch con los mensajes OFPT_SET_CONFIG y OFPT_GET_CONFIG_REQUEST, respectivamente. El switch responde a una solicitud de configuración con un mensaje OFPT_GET_CONFIG_REPLY; El switch no responde a un mensaje OFPT_SET_CONFIG, pero si observamos la secuencia de los mensajes en la Figura 4.28, vemos que inmediatamente después del mensaje OFPT_SET_CONFIG, el controlador envía al switch un mensaje OFPT BARRIER REQUEST, y como señala la especificación del protocolo, los mensajes ante una barrera deben procesarse por completo, incluido el envío de cualquier respuesta o error resultante, la barrera debe procesarse, y concluido esto el switch envía un mensaje OFPT BARRIER REPLAY. De esta forma el switch se configura conforme lo establecido por los mensajes OFPT_SET_CONFIG (Figura 4.33), OFPT BARRIER REQUEST (Figura 4.34), y la operación continuará luego del mensaje OFPT BARRIER REPLAY (Figura 4.35) enviado desde el switch. Ponga atención que el campo Transaction ID: de los mensajes BARRIER REQUEST/REPLAY tienen el mismo número, en el caso de nuestro ejemplo es 16, de esta forma queda asegurado tanto el orden como la ejecución de las acciones descritas en la barrera (BARRIER).

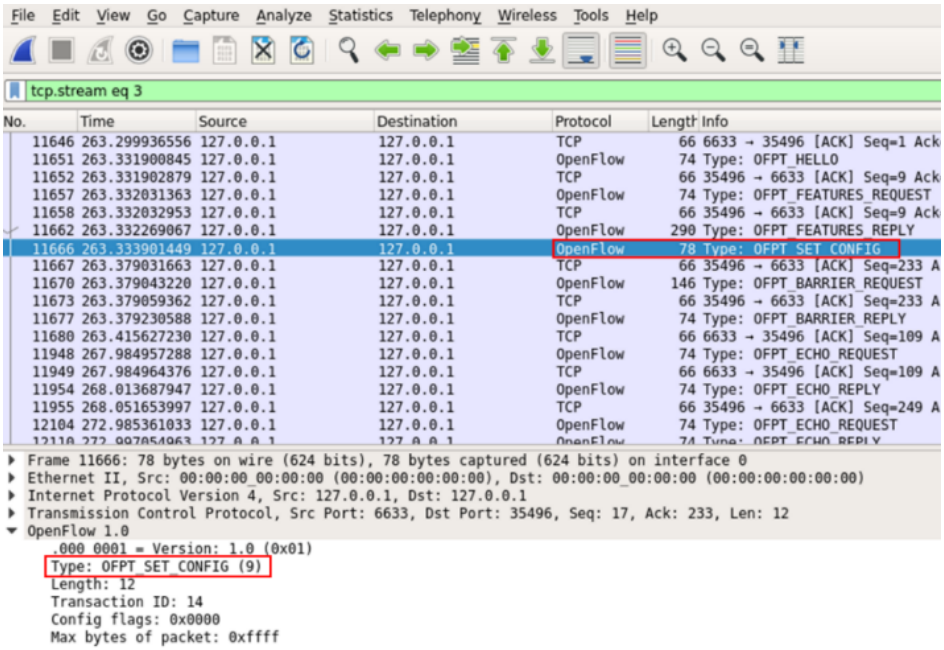

Figura 4.33: Mensaje OFPT SET CONFIG

La figura Figura 4.34 muestra que el mensaje de barrera incorpora una acción de (configuración), modificación en la tabla de flujo del switch OFPT_FLOW_MOD, y tiene al final del mensaje de barrera un campo Transaction ID: 16, hasta que no se cumpla con los establecido en OFPT_FLOW_MOD, (es decir, que se procese la barrera) no se recibirá el mensaje OFPT_BARRIER_REPLY con el mismo Transaction ID: 16 desde el switch. En general utilizando el mecanismo de barrera es como se asegura mantener consistentes las tablas de flujo en switches y cuando estas sean modificadas desde el

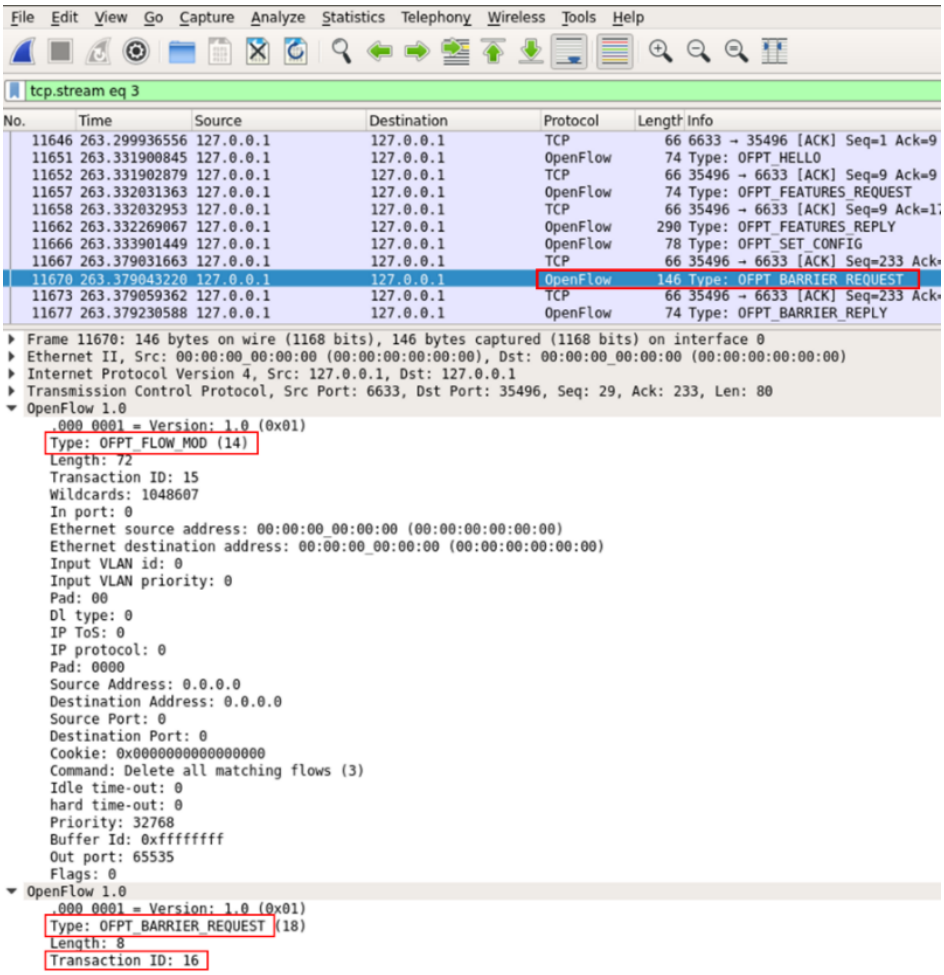

Figura 4.34: Mensaje OFPT BARRIER REPLAY

controlador, los mensajes se procesen en el orden adecuado.

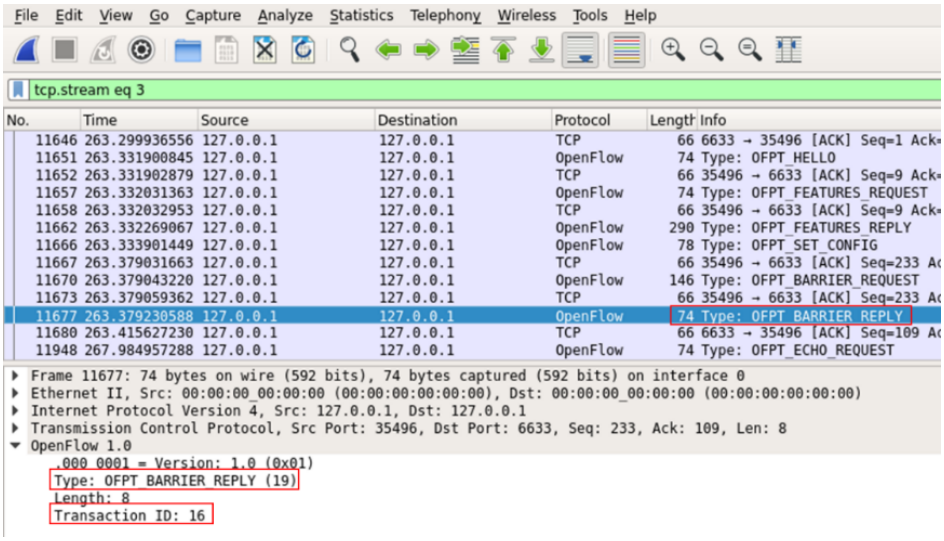

Figura 4.35

4.5.3. Obteniendo información de switches / datapaths desde línea de comandos

En la documentación de los comandos ovs-ofctl [94] y ovs-vsctl [95], se indica cómo se puede utilizar estos comandos en distintos contextos, para administrar tanto los switches como sus tablas de flujos, desde el shell de la MV Mininet, o bien desde una terminal conectada al dispositivo.

4.5.3.1. Comando ovs-vsctl show

Este comando muestra información sobre los puertos e interfaces con las que están conectados los switches y el controlador. Figura 4.36

Como se observa en la figura el comando entrega información muy útil como ser el nombre de los puertos / interfaces para poder identificarlos en las simulaciones y también el puerto donde atiende las peticiones tanto el controlador como el switch, en este ejemplo vemos que el switch s2 está conectado al controlador (is connected: true), y atiende las peticiones en el puerto 6655.

4.5.3.2. Comando ovs-ofctl show

Teniendo en cuenta la información obtenida por ovs-vsctl show, podemos solicitar a un switch específico sus características, mediante el comando ovs-ofctl show sX, donde X es el número que identifica al swith o datapath, en el ejemplo que muestra la Figura 4.37, es el switch s2 por ende, si ejecutamos, sudo ovs-ofctl show s2, desde el Shell de la MV Mininet, obtenemos la salida de la Figura 4 37. Como podrá advertir el lector esto coincide totalmente con la respuesta que entrega el switch al controlador ante una petición OFPT_GET_CONFIG_REQUEST.

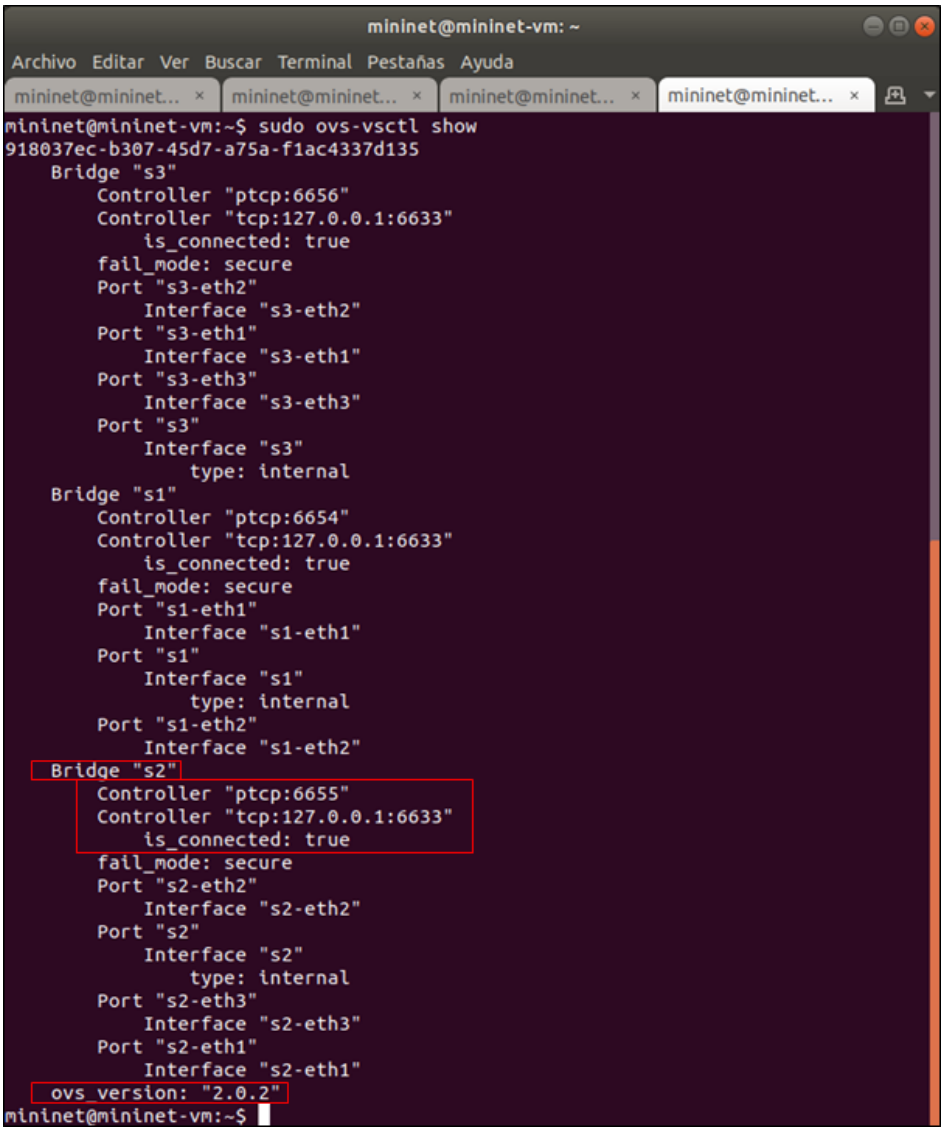

Figura 4.36: Comando ovs-vsctl show

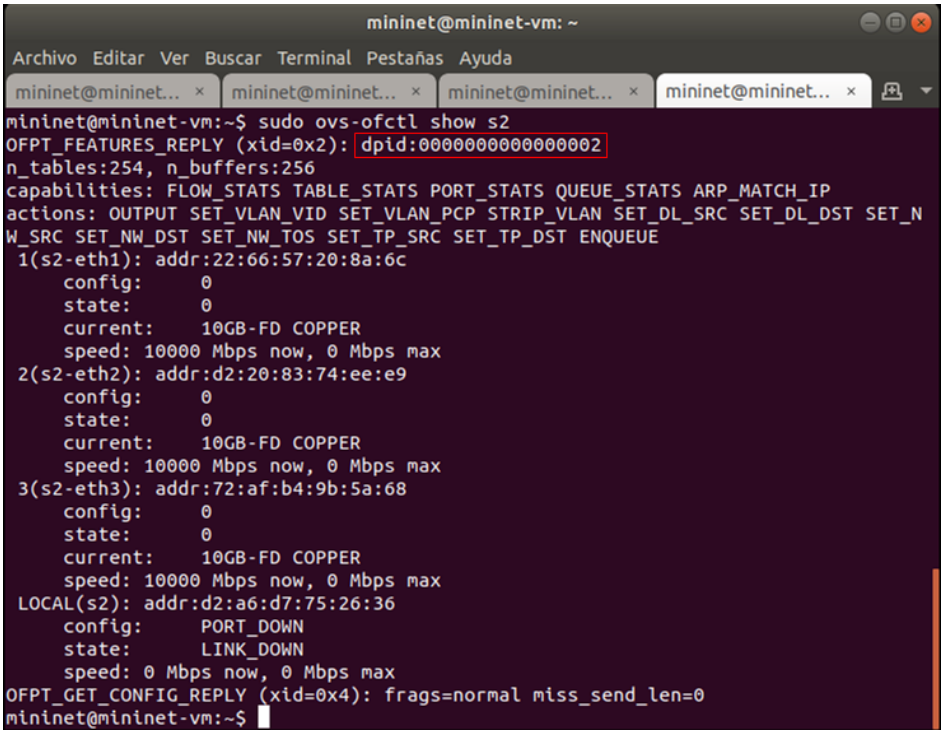

Figura 4.37: Comando ovs-ofctl show s2

4.5.3.3. Comando dpctl show

Otra forma de acceder a información del datapath específico es mediante el comando dpctl show sX tcp: ip controlador: puerto del switch, donde X es el número que identifica al datapath (tcp es el método de conexión que sabemos puede ser TLS o TCP ver capítulo 2 punto 2.10.1), la Figura 4.38 muestra la ejecución de dpctl show tcp:127.0.0.1:6655, el puerto 6655 como sabemos es donde atiende peticiones el switch que venimos usando de ejemplo.

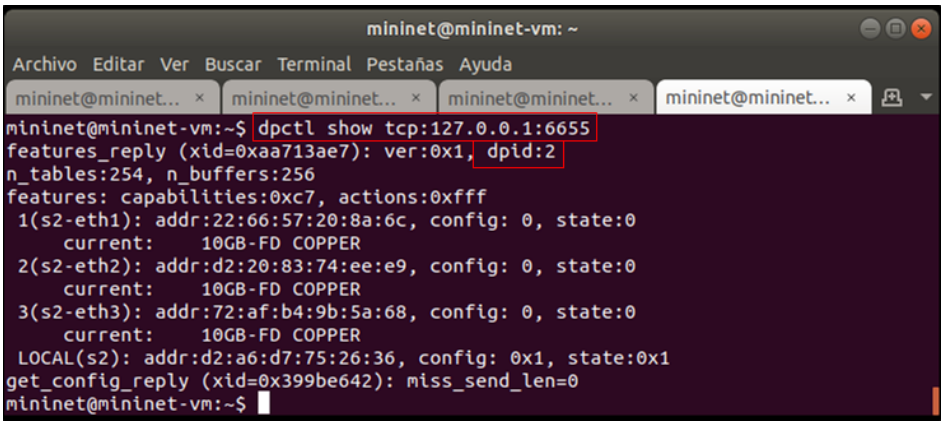

Figura 4.38: Comando dpctl show al datapath 2

4.5.4. Iniciando tráfico de paquetes

Cuando un paquete que llega al switch, si no encuentra ninguna coincidencia en ninguna de las entradas de la tabla correspondiente Un se debe crear un nuevo flujo. El switch debería tener configurado un descarte (Drop) de paquetes para un flujo que no haya sido definido con una entrada, pero sucede que, en la mayoría de los casos, el paquete será enviado al controlador mediante un mensaje Packet-in. El controlador entonces define un nuevo flujo para ese paquete y crea una o más entradas para la tabla. A continuación, el controlador envía la entrada o entradas al switch para que sean añadidas a sus tablas de flujo mediante un mensaje Packet-out. el cual contiene el paquete original, el mismo buffer_id que contenía el Packet-in y una acción que indica lo que hará con el paquete. Para poder estudiar este mecanismo en detalle, seguimos trabajando con la misma topología de la experiencia anterior y se procede a efectuar una captura con wireshark (no olvidar tener activo el filtro openflow_v1). Se procede en el siguiente orden (Figura 4.39): ping desde el host 1 (conectado al switch 1), al host 4 (conectado al switch2).

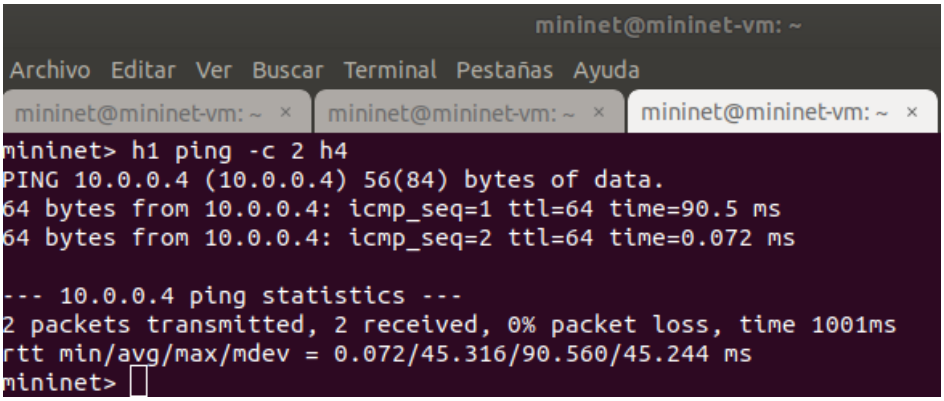

Figura 4.39: Ping desde h1 hacia h4

Como se describió en 4.5.1.1, el switch implementado en POX, es del tipo autoaprendizaje e instala reglas basadas únicamente en direcciones MAC, como muestra la Figura 4.40. ***Tip:*** *Aquí se puede*

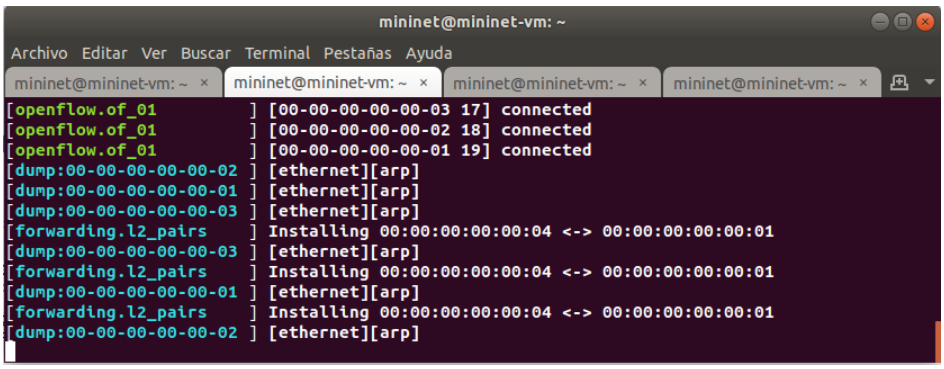

Figura 4.40: Controlador POX instalando reglas en los tres switches

apreciar la ventaja de utilizar la opción "–mac" (ver 4.4.3.2), al establecer la topología en Mininet, pues la misma identifica al host por la terminación, así fácilmente reconocemos al host 1 y al host 4 En la captura de la Figura 4.41 se pueden ver los eventos a partir de la llegada de un paquete al

switch, los cuales iremos describiendo en detalle.

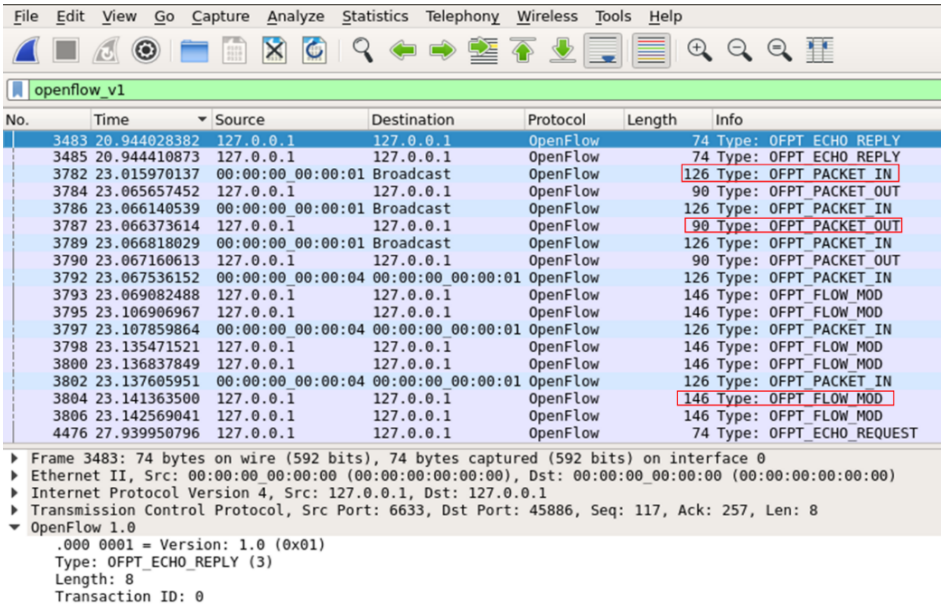

Figura 4.41: Processo de Packet-in y modificación de flujos

Los mensajes OFPT_ PACKET_IN, OFPT_PACKET_OUT y OFPT_FLOW_MOD, (recuadro rojo), los vemos repetidos 6 veces y esto es porque hay un camino de ida y otro de vuelta al ejecutar el ping, como son tres switches los que atraviesa el paquete ping, tenemos una ida y una vuelta por cada uno.

4.5.4.1. Mensaje OFPT_ PACKET_IN

En la Figura 4.42 se puede ver el detalle del mensaje OFPT_ PACKET_IN, producido en el switch 2, al ingresar el paquete ping originado en el host 1 y con destino al host 4, el paquete no encuentra ninguna coincidencia en la tabla de flujos del switch, pues al ser del tipo autoaprendizaje, no tiene definido ninguno aún.

La especificación respecto del ingreso de un mensaje Packet-in dice, si una entrada Table-Miss envía directamente paquetes al controlador utilizando el puerto CONTROLLER (ver [7] 4.5), el Packet-in debe identificar la razón como una entrada Table-Miss (flujo no identificado) (ver [7] subsección A.4.1). Lo podemos verificar en la Figura 4.42 en el campo Reason. Los valores posibles en el campo Reason son: • OFPR_NO_MATCH = 0, (No matching flow (table-miss flow entry). • OFPR_ACTION = 1, (Action explicitly output to controller). • OFPR_INVALID_TTL = 2, (Packet has invalid TTL) En el caso de nuestra captura el campo Reason tiene código (0) Asimismo, en la Figura 4.42, se observa que el paquete transporta una solicitud arp, con el fin de identificar la MAC de destino (host 4 con dirección IP =10.0.0.4)

Para completar el análisis del mensaje OFPT_ PACKET_IN, se mencionan algunos detalles que están incluidos en la especificación [7] en el punto A.4.1 Packet-In Message • El Buffer_id es un valor opaco utilizado por el datapath para identificar un paquete almacenado. Cuando se almacena un paquete, se incluirá una cantidad de bytes del mensaje en la parte de datos del mensaje. Si el paquete se envía debido a una acción de "send to controller", entonces se envían los bytes max_len desde el

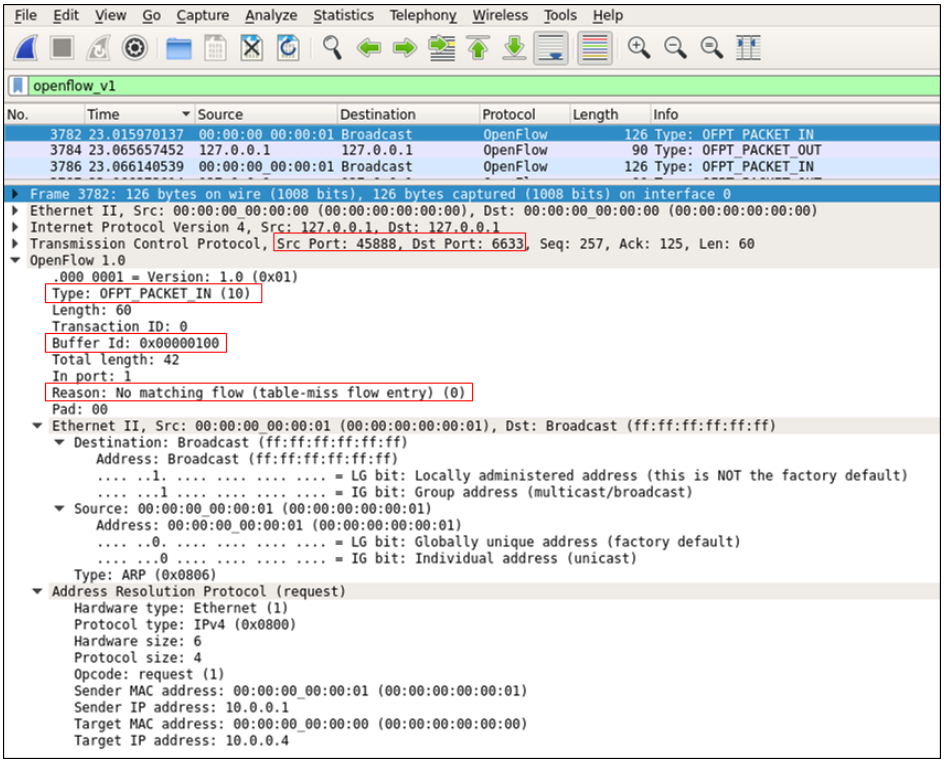

Figura 4.42: Detalle de mensaje OFPT_PACKET_IN

ofp_action_output de la solicitud de configuración de ujo. Si el paquete se envía por otros motivos, como un TTL no válido, se envían al menos bytes miss_send_len del mensaje OFPT_SET_CONFIG. El valor predeterminado miss_send_len es de 128 bytes. Si el paquete no está almacenado, ya sea debido a que no hay buffers disponibles, o debido a una solicitud explícita a través de OFPCML_NO_BUFFER, todo el paquete se incluye en la parte de datos, y el buffer_id es OFP_NO_BUFFER. Se espera que los switches que implementan la creación de buffers expongan, a través de la documentación, tanto la cantidad de buffers disponibles como la cantidad de tiempo antes de que se puedan reutilizar los mismos. Un interruptor debe manejar con mucho cuidado el caso en el que un mensaje de paquete empaquetado no produce ninguna respuesta del controlador. Un switch debe evitar que un buffer se vuelva a utilizar hasta que el controlador lo haya manejado, o que haya transcurrido una cierta cantidad de tiempo (indicado en la documentación). El campo de datos contiene el paquete en sí, o una fracción del paquete si el paquete está almacenado. El encabezado del paquete refleja cualquier cambio aplicado al paquete en el proceso anterior.

4.5.4.2. Mensaje OFPT_PACKET_OUT

Cuando el controlador desea enviar un paquete a través del datapath, utiliza el mensaje OFPT_PACKET_OUT

La Figura 4.43 muestra la captura del mensaje OFPT_PACKET_OUT, originado por el controlador con destino al switch 2. El buffer_id es el mismo que en el mensaje OFPT_ PACKET_IN. Esto asegura la integridad del paquete entrante y le indica al switch que hacer con el mismo, mediante el campo Actions, que en este caso le ordena reenviar el paquete. Para completar el análisis del mensaje OFPT_ PACKET_OUT, se mencionan algunos detalles que están incluidos en la especificación [7]

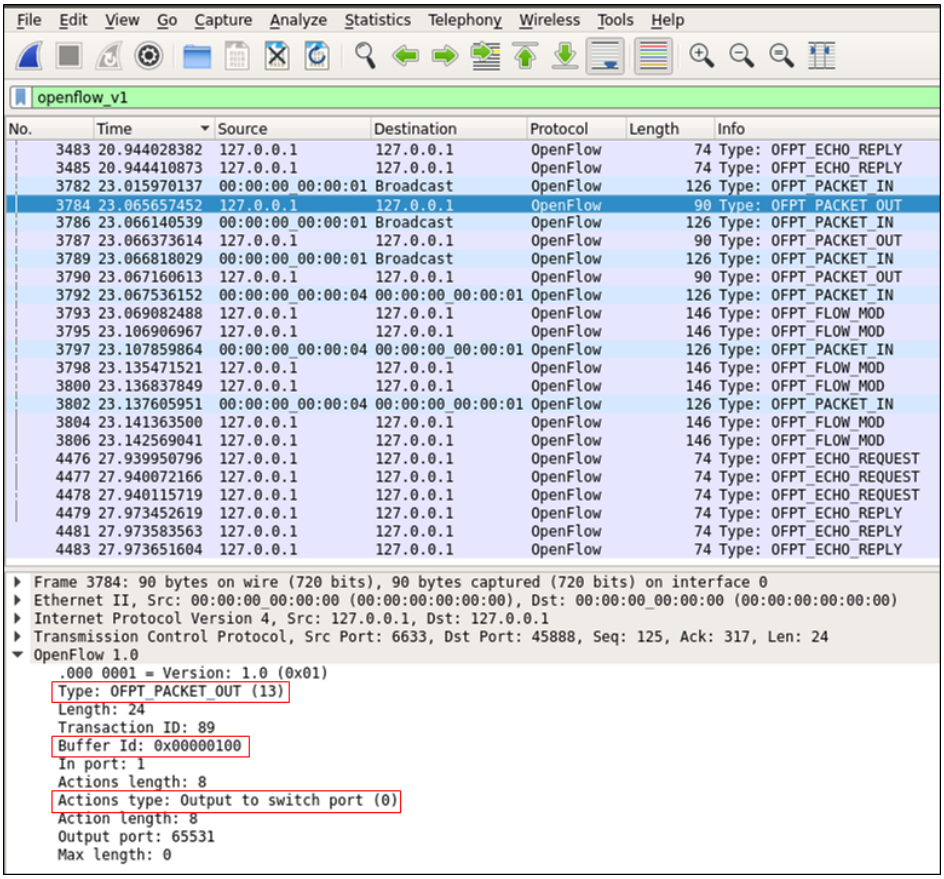

Figura 4.43: Mensaje OFPT_PACKET_OUT generado en el controlador

en el punto A.3.7 Packet-Out Message • El campo Actions es una lista de acciones que define cómo el switch debe procesar el paquete. Puede incluir modicación de paquetes, procesamiento de grupos y un puerto de salida. La lista de acciones de un mensaje OFPT_PACKET_OUT también puede especificar el puerto reservado OFPP_TABLE como una acción de salida para procesar el paquete a través de las entradas de ujo existentes, comenzando en la primera tabla de ujo. Si se especifica OFPP_TABLE, el campo in_port se usa como el puerto de ingreso en la búsqueda de la tabla de ujo. • Si el buffer_id es OFP_NO_BUFFER, entonces los datos del paquete son incluidos en la matriz de datos. OFP_NO_BUFFER es 0x . • En algunos casos, los paquetes enviados a OFPP_TABLE pueden reenviarse al controlador como resultado de una acción de entrada de ujo o una falla de la tabla. Detectar y tomar medidas para dichos bucles de controlador a switch está fuera del alcance de la especificación. En general, no se garantiza que los mensajes de OpenFlow se procesen en orden, por lo tanto, si un mensaje OFPT_PACKET_OUT que utiliza OFPP_TABLE depende de un flujo que se envió recientemente al switch (con un mensaje OFPT_FLOW_MOD), puede ser necesario un mensaje OFPT_BARRIER_REQUEST antes del mensaje OFPT_PACKET_OUT para asegurarse de que la entrada de flujo se confirmó en la tabla de flujo antes de la ejecución de OFPP_TABLE.

Como se mencionó anteriormente, los otros mensajes OFPT_PACKET_IN y OFPT_PACKET_OUT, que se observan en la figura de la captura, corresponden al camino de vuelta y únicamente varían en los puertos y las direcciones de origen y destino.

4.5.4.3. Mensaje OFPT_FLOW_MOD

Las modificaciones a una tabla de flujo desde el controlador se realizan con el mensaje OFPT_FLOW_MOD. En la Figura 4.44 se muestra la captura del mensaje OFPT_FLOW_MOD, consecuencia del mensaje OFPT_PACKET_OUT, que se genera en el controlador con destino al switch, para que este instale una entrada en su tabla para el flujo especificado

Como se observa en la Figura 4.44, el mensaje define todos los campos de coincidencia para instalar una entrada de flujo en la tabla del switch. A partir de este momento todos los paquetes que coincidan (matching), con esa entrada no necesitan ir al controlador, sino que son directamente conmutados con la información de la tabla.

Si bien el ejemplo obedece a la simulación con la topología que se viene trabajando, todos los detalles referentes a este tipo de mensaje pueden ser consultados en [7], sección A.3.4.1 Modify Flow Entry Message.

4.5.5. Trabajando con tablas de flujo en Mininet

Como se explicó en 4.5.1.2, se dispone de comandos que permiten administrar y/o configurar switches OpenFlow, a continuación, siguiendo con la misma topología de trabajo, se practicarán estos comandos mostrando sus correspondientes capturas.

4.5.5.1. "dpctl" en el entorno Mininet

El comando "dpctl", sirve para administrar datapaths (switchs) corriendo en el entorno Mininet sin la intervención del controlador, directamente al ejecutar el comando nos comunicamos con el switch o datapath implementado. Empecemos por ver los flujos instalados en las tablas de los switchs (datapaths) con el comando dpctl dump-flows (Figura 4.45).

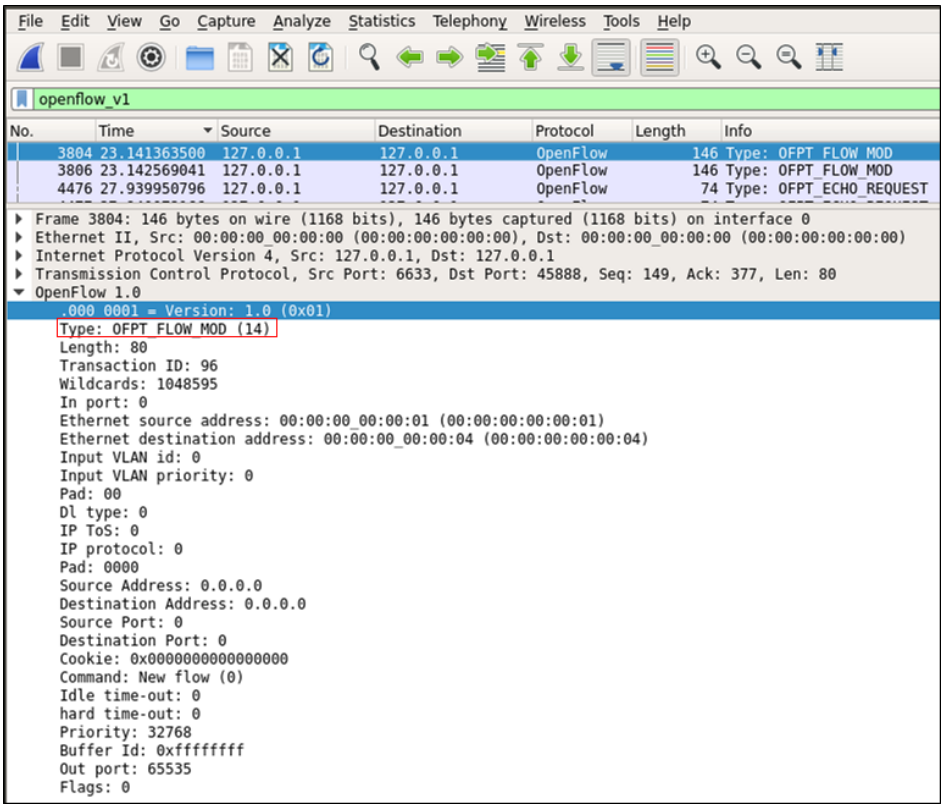

Figura 4.44: Mensaje OFPT_FLOW_MOD generado por el controlador con destino al switch 2

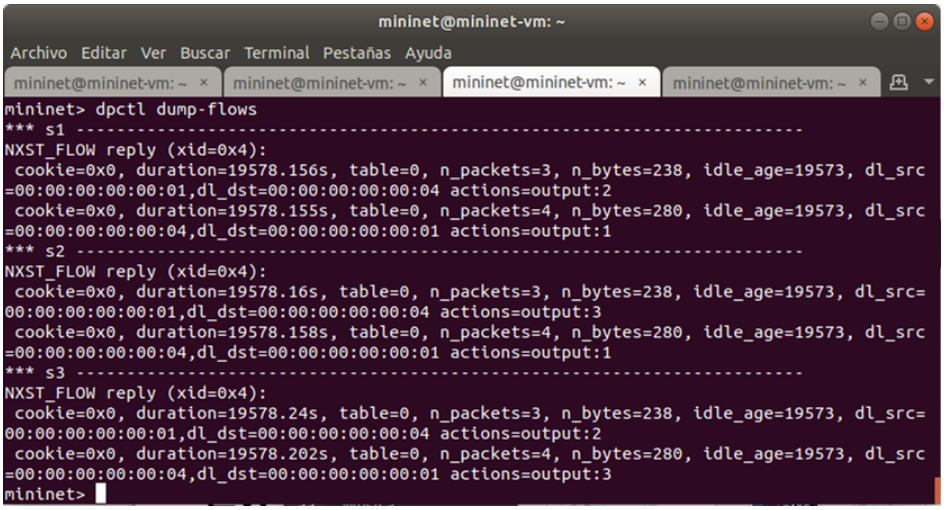

Figura 4.45: Comando dpctl en el entorno Mininet

Como se observa en la Figura 4.45, el comando dpctl dump-flows, muestra las entradas de la tabla para los flujos definidos en el switch, identificando los mismos como s1, s2 y s3. El comado datapath en particular forma parte de la simulación Mininet y se utiliza desde el cli (mininet>) de Mininet directamente.

4.5.5.2. Comando ovs-ofctl dump-flows

Como el switch reside en el kernel de SO, también puedo ocupar el comando ovs-ofctl dump-flows desde el promt de la MV Mininet como lo muestra la Figura 4.46, el comando se puede utilizar para visualizar la tabla de flujos de cada switch por separado.

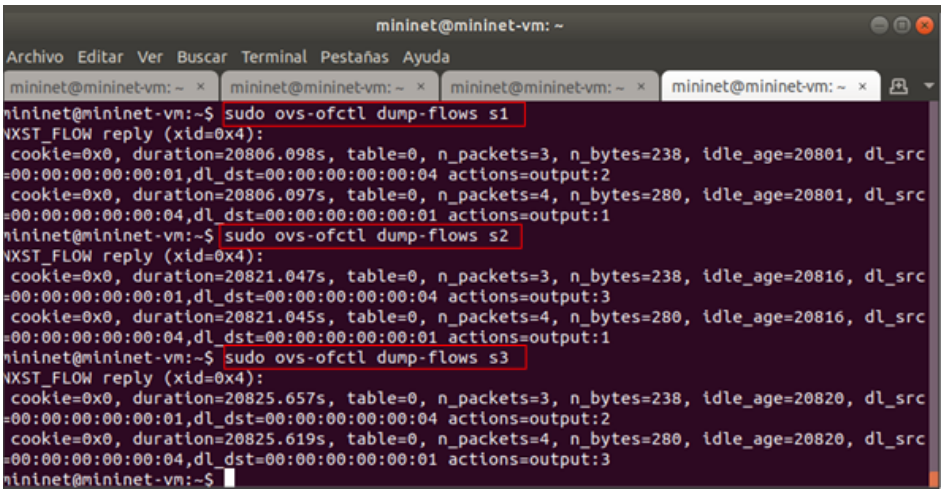

Figura 4.46: Comando ovs-ofctl desde el promt de la MV Mininet

4.5.5.3. comando ovs-dpctl dump-flows tcp:127.0.0.1: [puerto del proceso]

Otra forma de visualizar la información de la tabla desde el promt de la MV Mininet, es utilizar el comando ovs-dpctl dump-flows tcp:127.0.0.1:6654, Figura 4.47, preste atención que el puerto en donde está corriendo el proceso del switch, es el número siguiente de donde está el controlador, en este caso 6653, y de esta misma forma, invocando el mismo comando en los puertos 6655 y 6656, se puede ver las tablas de los respectivos switches. ***Tip:*** *Tenga presente lo comentado respecto de los puertos asignados a OpenFlow y el comentario que hicimos en 2.10.5, antes el puerto por defecto para el controlador era 6633, y en las actuales implementaciones es 6653, este pequeño detalle nos llevó a encontrarnos con que "no funcionaba el comando", para cuando en realidad la implementación ya había cambiado el puerto, (como comentario, esta anécdota nos llevó un considerable tiempo revisando que cosa es la que hacíamos mal ;))*

También es posible acceder a los comandos de shell externos desde el cli mininet>, utilizado sh, por ejemplo, podemos hacer lo mismo que muestra la Figura 4.44, pero sin salir del entorno de simulación donde tenemos ejecutando nuestra topología escribiendo el comando como sigue: • mininet>sh dpctl dump-flows tcp:127.0.0.1:6654 el resultado de esto lo podemos ver en la Figura 4.48:

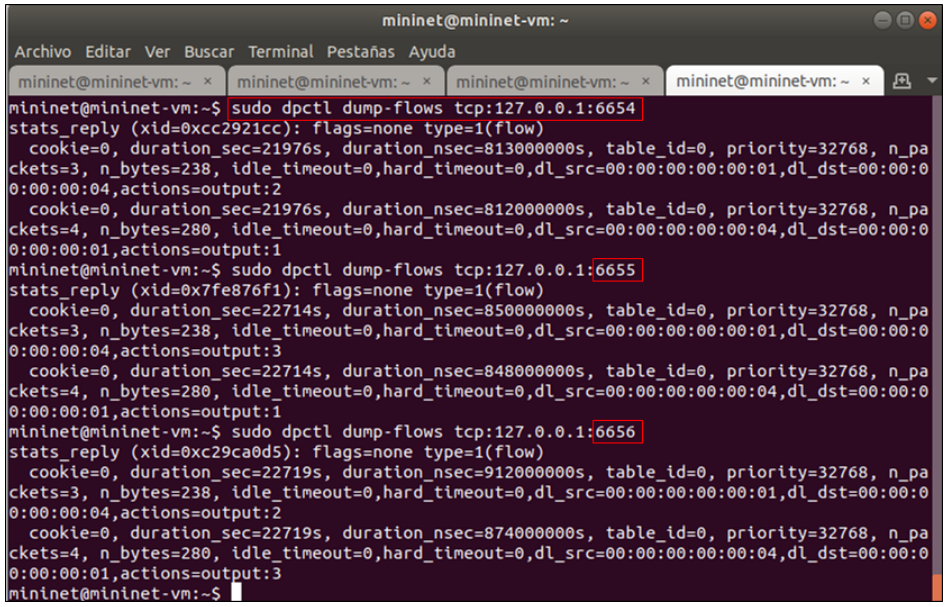

Figura 4.47: - comando ovs-show tcp:127.0.0.1:6654 , 6655 y 6656

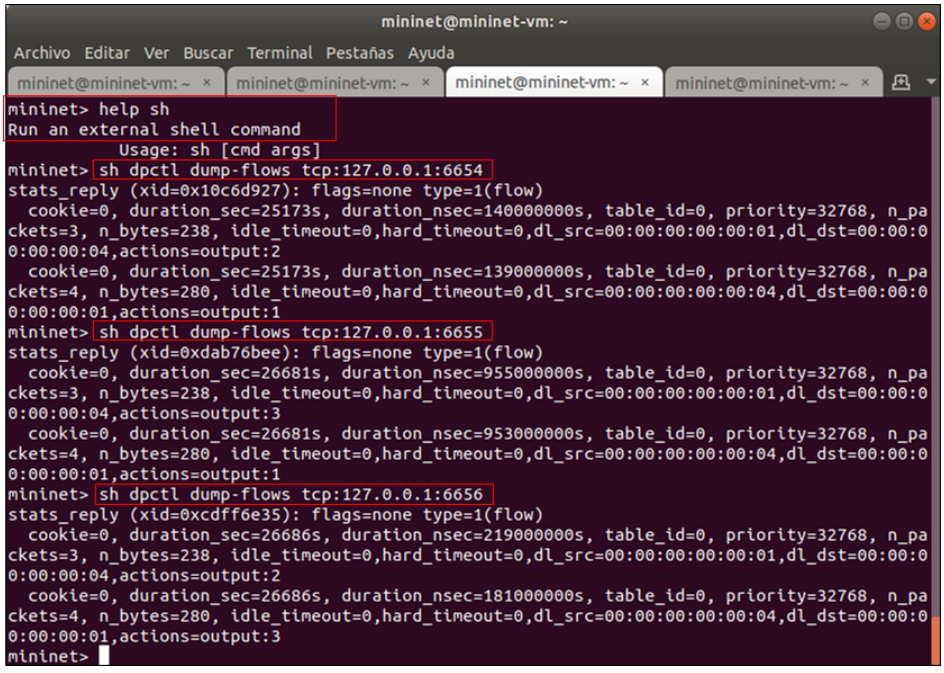

Figura 4.48: Comando sh dpctl dump-flows tcp:127.0.0.1:6654 desde el cli Mininet

4.5.6. Administrando Flujos con Mininet

En los párrafos anteriores, se vio como utilizar distintos comandos de línea para poder acceder a las tablas de los switches, a continuación, veremos cómo instalar, modificar o borrar entradas de flujo en las tablas. Como tenemos funcionando la red simulada (topología en árbol con tres switches y cuatro hosts), y vemos en las figuras 4-37 a 4-39, todos los flujos instalados. hacemos un pingall para verificar la conectividad entre todos los hosts, (Figura 4.49).

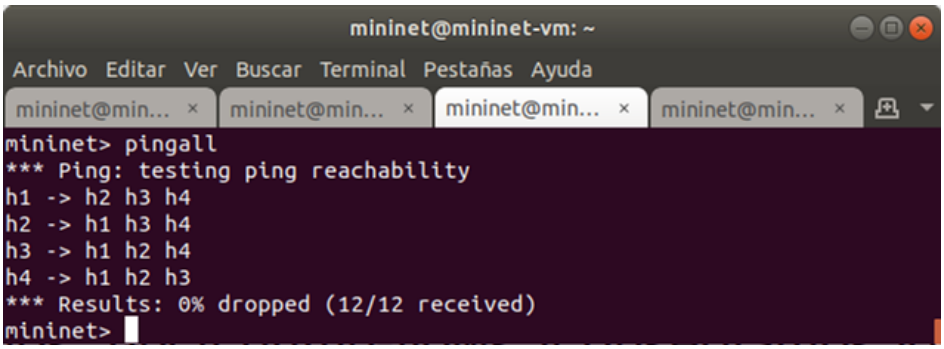

Figura 4.49: Comando pingall para verificar conectividad entre todos los hosts

Tal como muestra la Figura 4.49, tenemos total conectividad entre los hosts.

4.5.6.1. Comando dpctl del-flows

Si, por ejemplo, se desea borrar todas las entradas de flujos en las tablas, una forma sencilla de hacerlo es mediante el comando de cli mininet>dpctl del-flows. Figura 4.50.

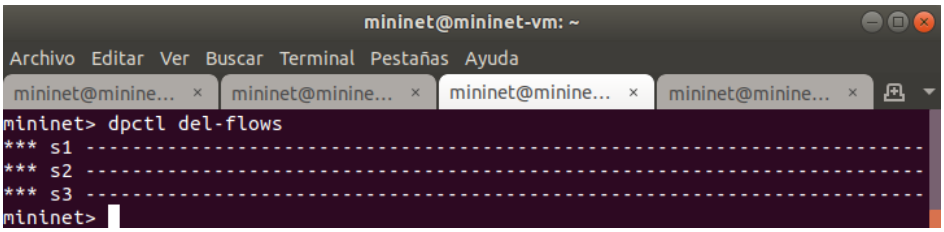

Figura 4.50: Comando dpctl del-flows

Verificamos viendo las tablas de flujo con dpctl dump-flows Figura 4.51.

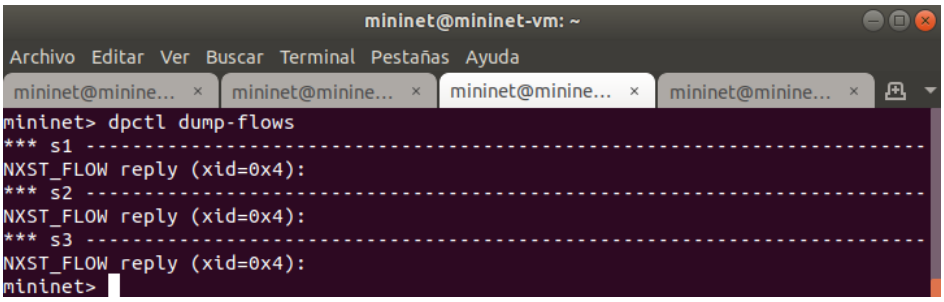

Figura 4.51: Verificando que se borraron todas las entradas de flujo en las tablas

En estas condiciones, los switches no tienen ninguna entrada en sus tablas (lo que significa que todo paquete entrante deberá hacer el proceso Packet-in y el controlador instalara la correspondiente entrada de flujos en la tabla pues como vimos los switches que estamos utilizando en esta experiencia son de auto aprendizaje basado en MAC). Ahora bien, que sucede si por ejemplo quisiera aislar al host 2 de manera tal que no pueda cursar ningún tráfico al resto de la red, esto se logra mediante una entrada en la tabla de flujo indicando, que todo lo que ingresa por el puerto correspondiente del switch2, y que coincida con la MAC del host 2 tenga como Actions=Drop (descarte). El procedimiento se muestra a continuación.

4.5.6.2. 4.5.6.2 Comando ovs-ofctl add-flow

Se aplica el comando desde el prompt de la MV Mininet, tal como muestra la Figura 4.52.

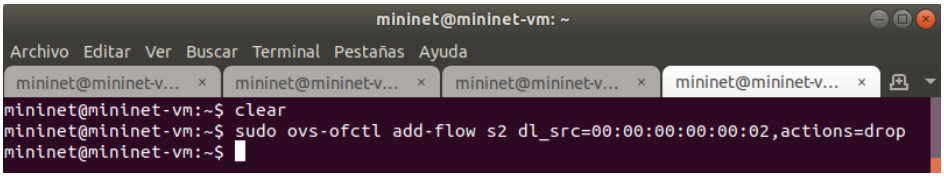

Figura 4.52: Comando ovs-ofctl add-flow

La Figura 4.53, muestra que el switch 2 tiene la entrada con campo de correspondencia MAC del host 2 y la acción definida es el descarte. Procedemos a verificar nuevamente la conectividad con pingall Figura 4.54:

Figura 4.53: Verificando como quedaron las tablas de flujos en los tres switches

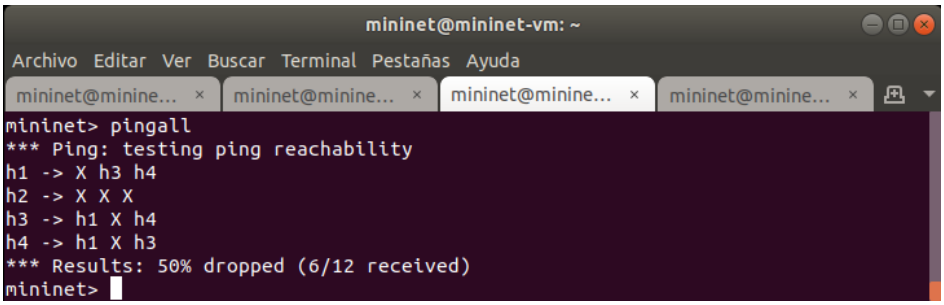

Figura 4.54: Controlando la conectividad entre los hosts, luego de modificar las tablas en los switches

Claramente se observa que los sitios correspondientes a h2, están marcados con X, esto significa que ahí no hay conectividad, verificando entonces la regla que introdujo la modificación de la tabla de s2 (Figura 4.53). Algo a destacar en el proceso que acabamos de realizar, es que las tablas después

del comando pingall, se verán modificadas por proceso de Packet-in de todos aquellos paquetes que no encuentran coincidencia en las mismas, pero para h2 existirá coincidencia, no habrá Packet-in para los paquetes que provengan de h2, sino directamente serán descartados, y esa entrada no es cambiada por el controlador. Lo podemos comprobar, haciendo un nuevo dump-flows como muestra la Figura 4.55.

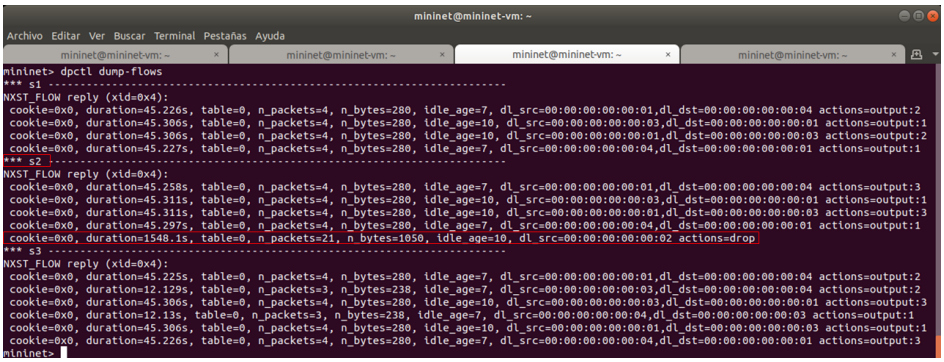

Figura 4.55: Tablas de los switches luego del pingall

4.6. MiniEdit, la interfaz gráfica de usuario de Mininet

4.6.1. Intérprete de Python

Si la primera frase en la línea de comandos de Mininiet es py, entonces ese comando se ejecuta con Python. Esto podría ser útil para extender Mininet, así como para probar su funcionamiento interno. Cada host, switch y controlador tiene un objeto Node asociado. Los nodos proporcionan una abstracción simple para interactuar con hosts, switches y controladores. Los nodos locales son simplemente uno o más procesos en la máquina local.

4.6.2. Interfaz de usuario MiniEdit

Un ejemplo de lo versátil y poderoso de esto; lo podemos ver en el script de Phyton "MiniEdit", El script "miniedit.py", se encuentra en la carpeta de ejemplos de Mininet. Para ejecutar MiniEdit, ejecute el comando: • $ sudo /mininet/examples/miniedit.py MiniEdit tiene una interfaz de usuario simple que presenta un área de trabajo, con una fila de iconos de herramientas en el lado izquierdo de la ventana y una barra de menú en la parte superior de la ventana (Figura 4.56).

La Figura 4.57 enumera las opciones más usadas y a continuación se detallan cada una de ellas.

- *La herramienta Seleccionar, se utiliza para mover nodos alrededor del área de trabajo. Haga clic y arrastre cualquier nodo existente. Curiosamente, la herramienta Seleccionar no es necesaria para seleccionar un nodo o enlace en el área de trabajo. Para seleccionar un nodo o enlace existente, simplemente desplace el puntero del ratón sobre él (esto funciona independientemente de la herramienta actualmente activa), y luego haga clic con el botón derecho para mostrar un menú de configuración para el elemento seleccionado o presione la tecla Eliminar para eliminar el elemento seleccionado.*

- *La herramienta Host crea nodos en el área de trabajo, estos nodos cumplen la función de las computadoras host. Haga clic en la herramienta, luego haga clic en cualquier lugar del área de*

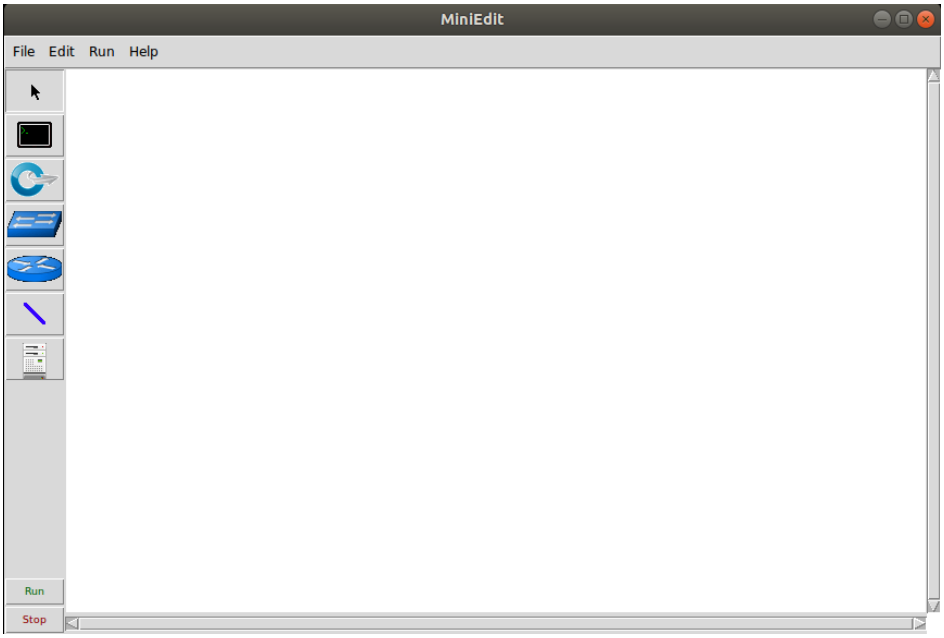

Figura 4.56: Interfaz gráfica de usuario MiniEdit

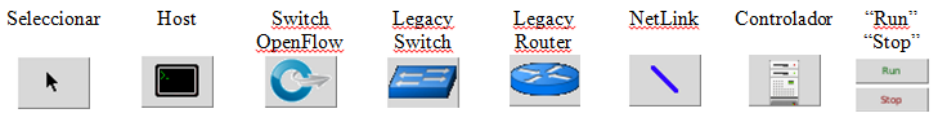

Figura 4.57: Opciones de la Interfaz gráfica de usuario MiniEdit

trabajo donde desee colocar un nodo. Mientras la herramienta permanezca seleccionada, puede seguir agregando hosts haciendo clic en cualquier lugar del área de trabajo. El usuario puede configurar cada host haciendo clic con el botón derecho en él y seleccionando Propiedades en el menú.

- *La herramienta Switch OpenFlow, crea switches OpenFlow en el área de trabajo. Se la herramienta Hosts mostrada en el párrafo anterior. El usuario puede configurar cada switch, haciendo clic con el botón derecho sobre él y seleccionando Propiedades en el menú.*

- *La herramienta Legacy Switch crea un switch Ethernet de autoaprendizaje con la configuración predeterminada. El Legacy Switch funcionará independientemente sin un controlador; no se puede configurar y está configurado por defecto con Spanning Tree deshabilitado, por lo tanto, no conecte los Legacy Switch en un bucle.*

- *La herramienta Legacy Router, crea un enrutador básico que funcionará de manera independiente, sin un controlador. Básicamente es solo un host espera que estos switches estén conectados a un controlador. La herramienta funciona de la misma manera que con IP Forwarding habilitado. El Legacy Router, no se puede configurar desde la GUI de MiniEdit*

- *La herramienta NetLink crea enlaces entre nodos en el área de trabajo. Cree enlaces seleccionando la herramienta NetLink, luego haga clic en un nodo y arrastre el enlace al nodo objetivo. El usuario puede configurar las propiedades de cada enlace haciendo clic derecho en él y seleccionando Propiedades en el menú.*

- *La herramienta Controlador crea un controlador. Se pueden agregar múltiples controladores. De forma predeterminada, el MiniEdit crea un controlador de referencia de OpenFlow de Mininet, el cual implementa el comportamiento de un interruptor de aprendizaje (visto en el punto 4.4.2). Se pueden configurar otros tipos de controladores. El usuario puede configurar las propiedades de cada controlador haciendo clic derecho en él y seleccionando Propiedades en el menú.*

- *El botón "Run", arranca la topología de simulación de Mininet que se muestra actualmente en el área de trabajo de MiniEdit. El botón "Stop", lo detiene. Cuando la simulación de MiniEdit se encuentra en el estado .Ejecutar", hacer clic con el botón derecho en los elementos de la red revela funciones operativas, como abrir una ventana de terminal, ver la configuración del interruptor o configurar el estado de un enlace en üp.º "down".*

4.6.2.1. Menú Edit – Preferencias

Antes de arrancar una simulación al utilizar MiniEdit, es conveniente fijar las preferencias con el Menú Edit, tal como muestra la Figura 4.58.

Lo primero que aparece, es la casilla que permite fijar el prefijo IP para los hosts de la red de simulación. Note, que es muy importante dejar tildada la casilla de verificación "Start CLI", pues esto permitirá interactuar con la topología de simulación, dado que nos habilita el uso del CLI de Mininet. El botón "Default Terminal", permite escoger entre Xterm o Gterm. El botón "Defaul Switch", permite seleccionar el tipo de switch que estará disponible en la herramienta "Switch OpenFlow", Otro detalle interesante de mencionar, es que se puede fijar el puerto de escucha del Datapath (recordar que por

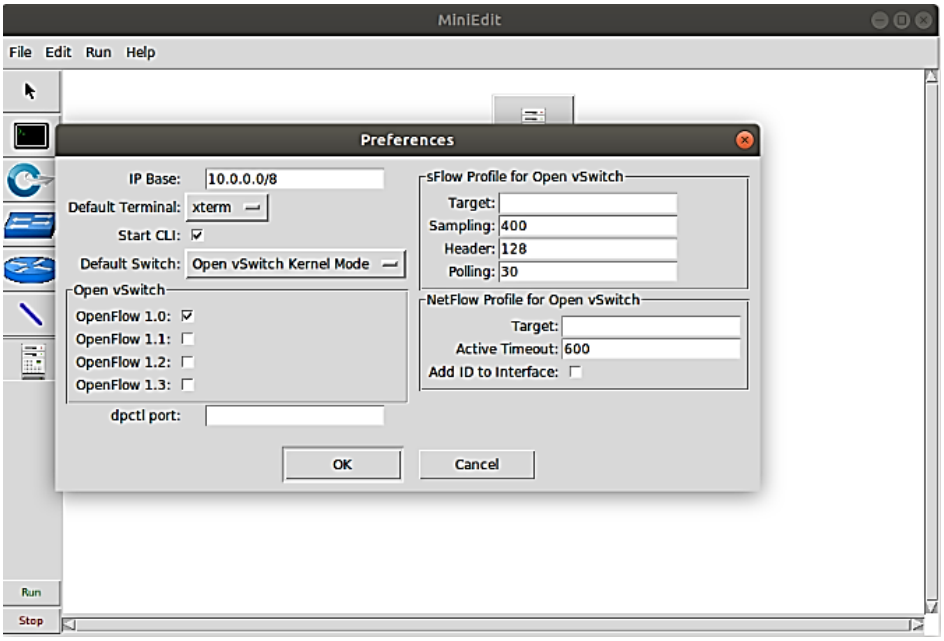

Figura 4.58: Ajuste de preferencias en MiniEdit

defecto siguen un orden consecutivo a partir del puerto del controlador), colocando el valor en la casilla "dpctl port".

También es posible seleccionar la versión del protocolo OpenFlow a utilizar, en el caso de la Figura 4 57, está seleccionada la casilla de verificación OpenFlow 1.0, puesto que en la topología de demostración estamos trabajando con un controlador POX que solo soporta esta versión, no obstante, dependiendo del controlador y los switches se pueden escoger las otras versiones en forma aislada o simultánea.

4.6.2.2. Configurando parámetros del controlador

La figura Figura 4.59, muestra las opciones de configuración que nos ofrece la herramienta crear Controlador. Permite, colocar el nombre con que aparece en los gráficos, fijar el puerto de escucha, definir el tipo de controlador de entre cuatro posibilidades, i) Remote Controller ii) In-Band Controller iii) OpenFlow Reference iv) OVS Controller. Para el caso de Remote Controller e In-Band Controler, se puede colocar la dirección IP del controlador. El otro botón de configuración es "Protocol", y permite definir si el canal OpenFlow del controlador, va a utilizar el protocolo TCP o SSL.

De la misma forma y como se expuso en la descripción inicial, cada elemento se puede configurar haciendo un clic con el botón derecho del ratón y eligiendo propiedades como se muestra a continuación.

4.6.2.3. Configurando parámetros del switch OpenFlow

La Figura 4.60, muestra las opciones de configuración (botón derecho del ratón) de la herramienta Switch OpenFlow. Es interesante destacar que el botón Switch Type permite elegir entre cuatro tipos: i) Default (switch de referencia OpenFlow) ii) Open vSwitch Kernel Mode iii) Userspace Switch iv) Userspace Switch inNamespace. Una vez que se seleccionó el tipo de switch, permite fijar la dirección IP y puerto de escucha Datapath. Los comandos de arranque / parada en la parte inferior y por último a la izquierda la posibilidad de agregar interfaces externas

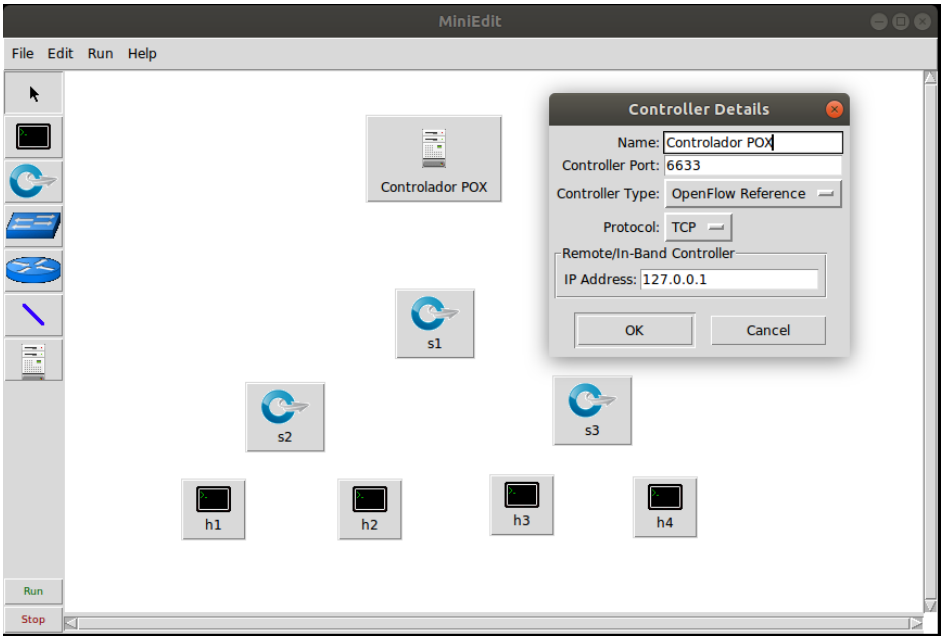

Figura 4.59: Fijando los detalles del controlador en MiniEdit

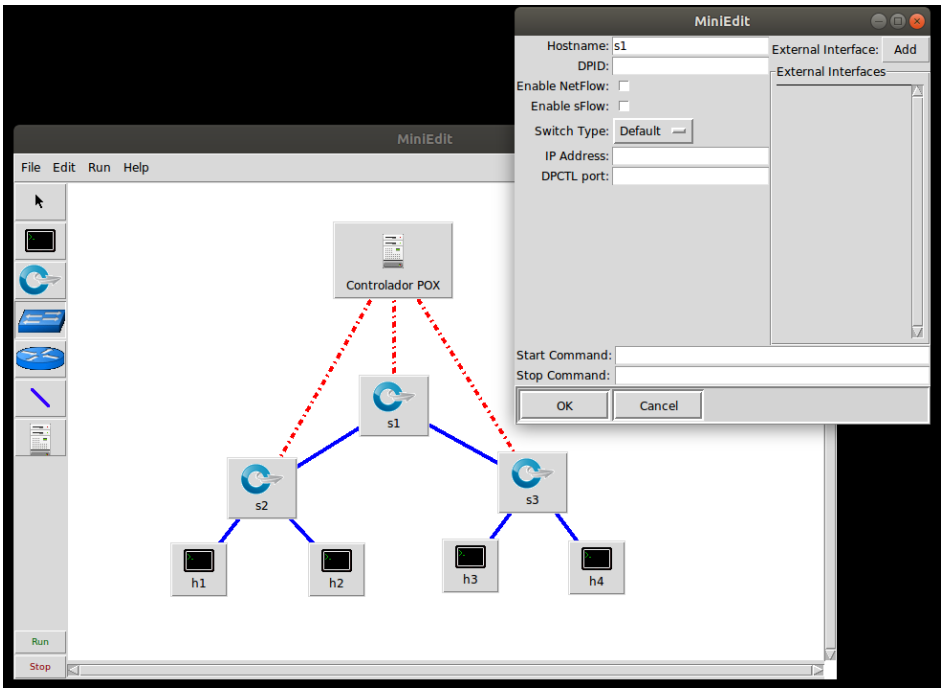

Figura 4.60: Fijando propiedades del Switch OpenFlow en MiniEdit

Con relación a las interfaces externas, si se desea añadir las interfaces de red de la máquina virtual que van a conectarse con las otras máquinas virtuales que actúan como equipos finales a la instancia de Open vSwitch que se ha creado (Figura 4.61), escribir el nombre de la interfaz "física", en el espacio que se genera con el botón "Add", equivale a la instrucción de línea de comando:

mininet@mininet-vm $ sudo ovs-vsctl add-port s1 eth2.

Es de destacar, que se pueden agregar distintas interfaces pulsando repetidamente el botón "Add".

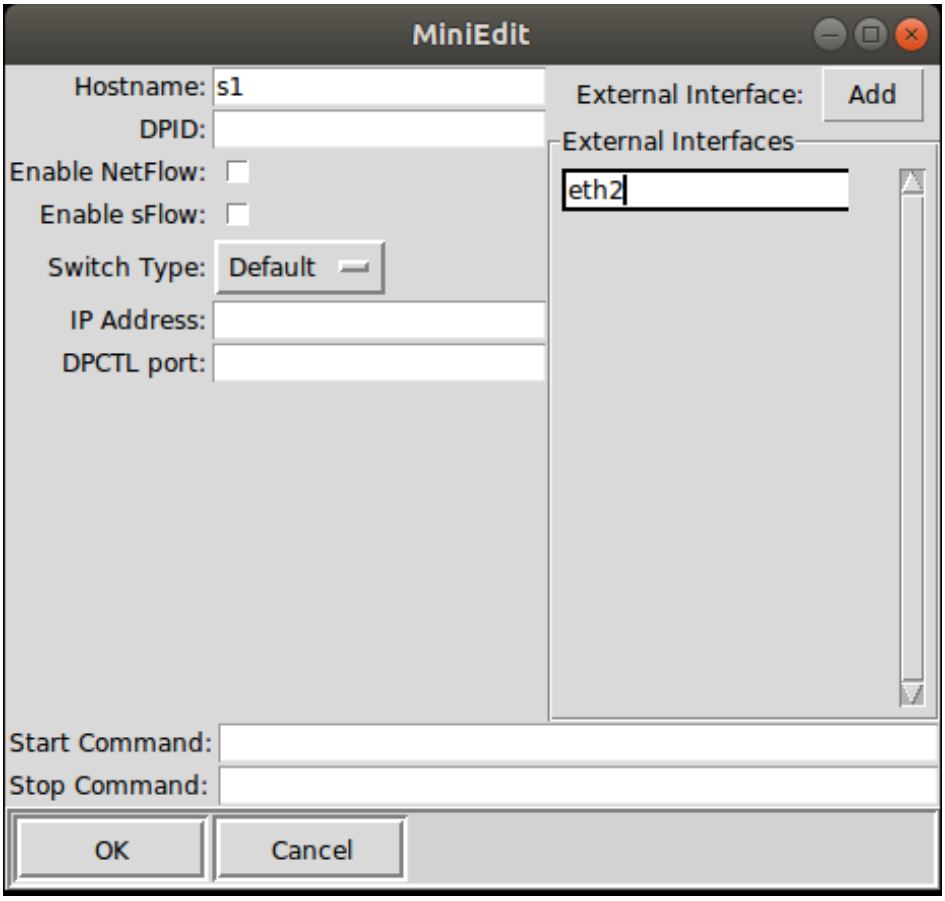

Figura 4.61: Añadiendo interfaz externa a Switch OpenFlow con MiniEdit

Es importante entender este paso. Lo que se acaba de hacer es asociar la interfaz "física" eth2, de la máquina virtual, con la instancia de OVS "s1". Esto permitirá que todo el tráfico que llegue por dicha interfaz de la MV (es decir, el tráfico proveniente de los equipos finales) sea conmutado por la instancia de OVS "s1" que se ejecuta en su interior. El último paso es asignarles direcciones IP a las interfaces de los equipos finales. Estas direcciones deben pertenecer al mismo segmento de red.

4.6.2.4. Configurando parámetros de los enlaces

La Figura 4.62 muestra el detalle de configuración de los parámetros de los enlaces, esto tiene directa relación a lo expuesto en el punto 4.4.3.3, en los enlaces emulados, la velocidad de datos de cada enlace es impuesta por Linux Traffic Control (tc), que tiene varios programadores de paquetes para configurar el tráfico a una velocidad configurada (ver 4.4.3.3.1). Si bien no es posible trabajar con toda la variedad de parámetros existentes mediante "tc", la interfaz gráfica permite configurar los

más ubicuos en la mayoría de las simulaciones, no obstante, en necesario recordar que esta aplicación tiene la posibilidad de generar el script de la topología estudiada mediante el Menú File >Export Level 2 Script, (esto se muestra más adelante en el punto 4.6.2.6) y editándolo se puede agregar mayor nivel de detalle a los parámetros.

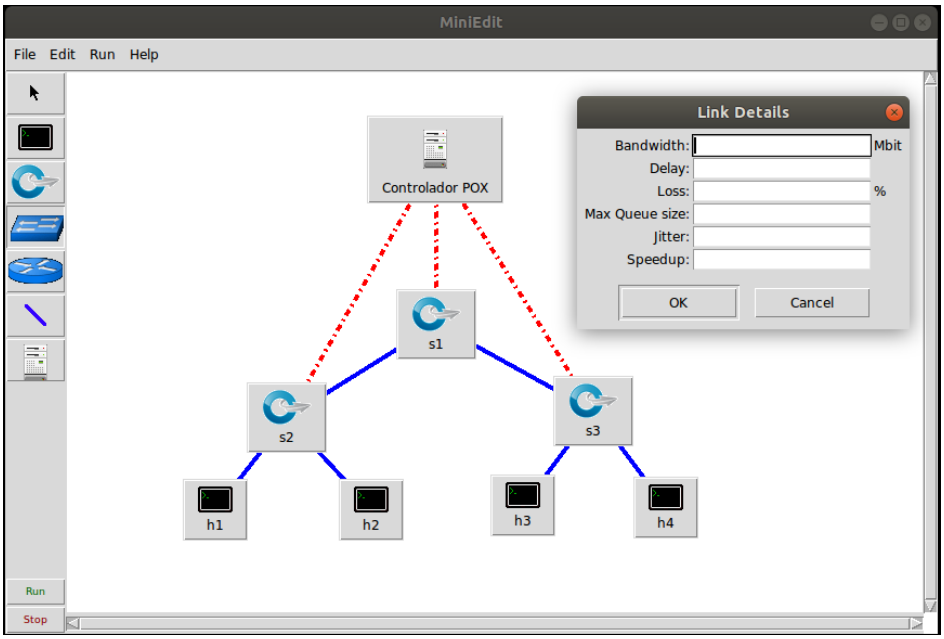

Figura 4.62: Parámetros de enlaces en MiniEdit

4.6.2.5. Configurando parámetros de hosts

Cada host emulado tiene sus propias interfaces virtuales de Ethernet (creadas e instaladas con ip link add / set). Un par de Ethernet virtual (o veth), actúa como un cable que conecta dos interfaces virtuales o puertos de conmutador virtual; los paquetes enviados a través de una interfaz se entregan a la otra, y cada interfaz aparece como un puerto Ethernet completamente funcional para todos los sistemas y aplicaciones de software. Las figuras Figura 4.63, Figura 4.64, Figura 4.65 y Figura 4.66, muestran las posibilidades de configuración que ofrece la interfaz gráfica.

4.6.2.6. Menú File en MiniEdit

El menú File, contiene las opciones para abrir / guardar (Open / Save), las topologías creadas con la interfaz de usuario gráfica. Para el caso de guardar (Save), se puede hacer con la extensión propia de la aplicación (.mn) mediante la opción Menú File Save, o bien utilizar la opción Menú File >Export Level 2 Script, que genera y guarda el script que ejecuta la topología creada con la interfaz de usuario gráfica, con la extensión (*.py). esto último tiene dos ventajas importantes:*

- *i) Se puede ejecutar la topología directamente desde la línea de comando, por ejemplo, si se exporto el script Level2 al directorio /Mininet/custom, de la MV Mininet, entonces se puede ejecutar simplemente con: $ sudo python /mininet/custom/"nombre de archivo".py*

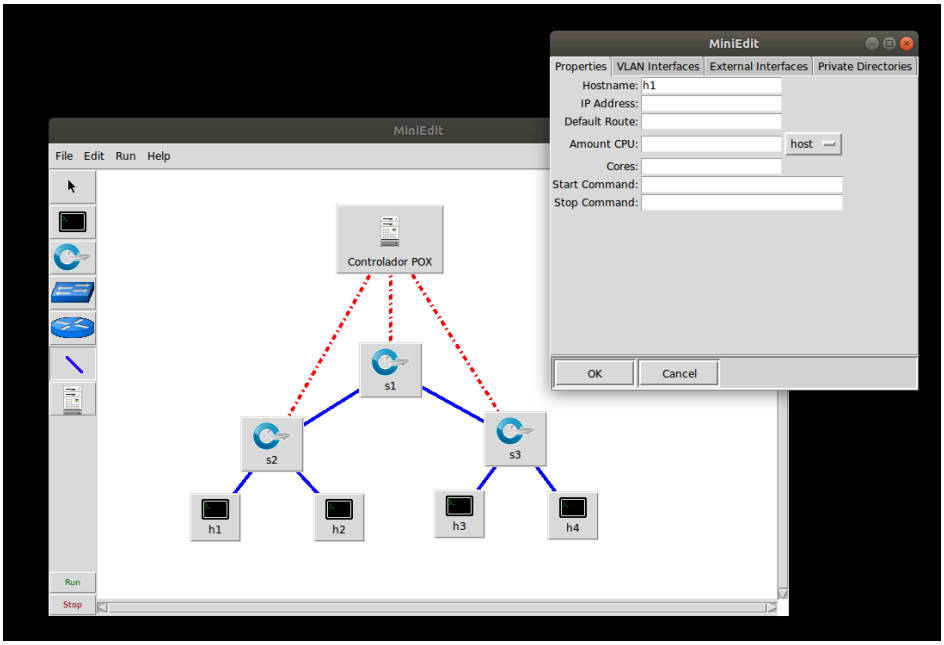

Figura 4.63: Parámetros de los hosts en MiniEdit

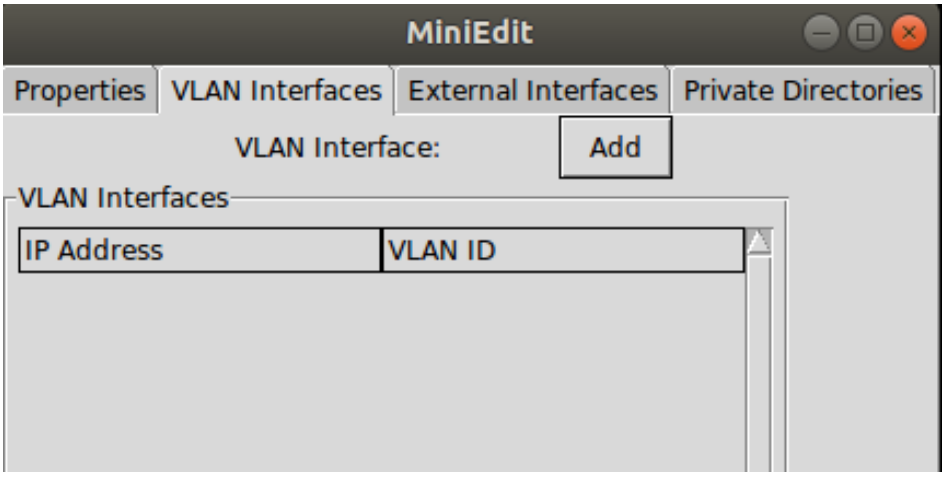

Figura 4.64: Detalle configuración VLAN en un host con MiniEdit

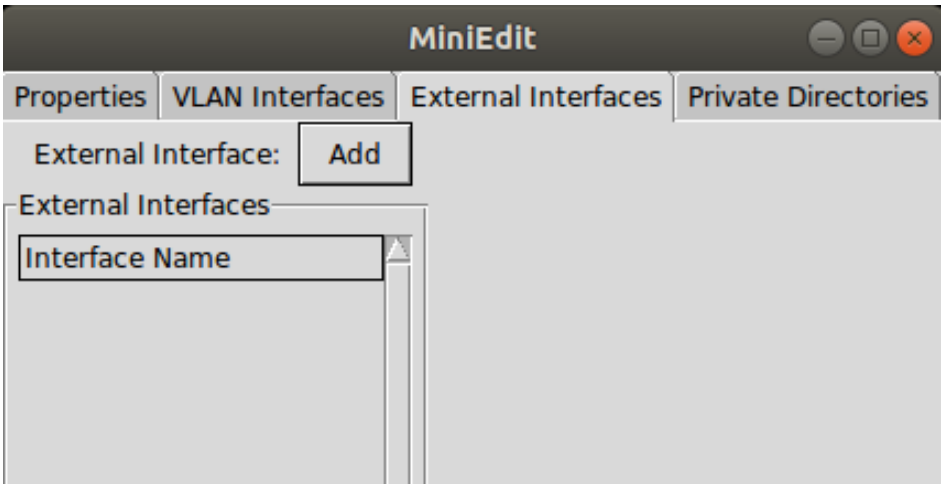

Figura 4.65: Detalle configuración interfaz externa en un host con MiniEdit

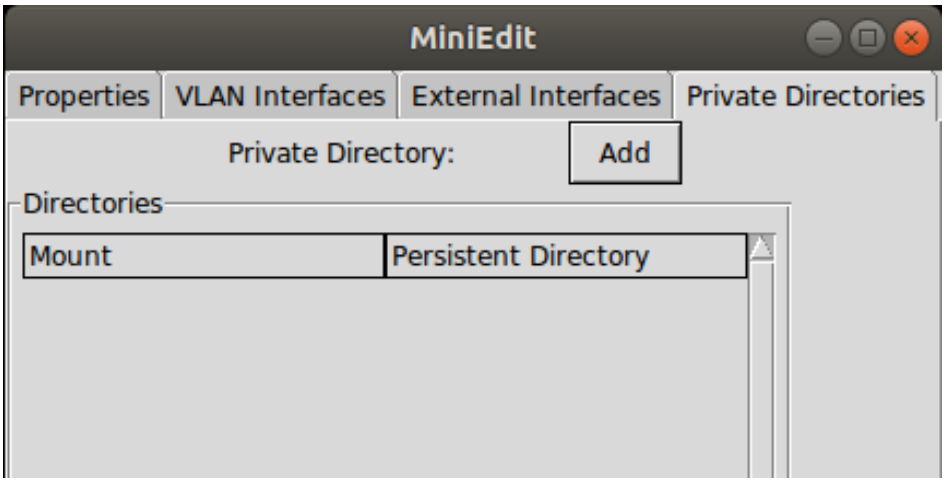

Figura 4.66: Detalle configuración de directorios en un host con MiniEdit

- *ii) Teniendo la script guardada, se puede editar para agregar elemento y/o complejidad al escenario de simulación, un ejemplo típico sería agregar parámetros a los enlaces de la topología. (Figura 4.67)*

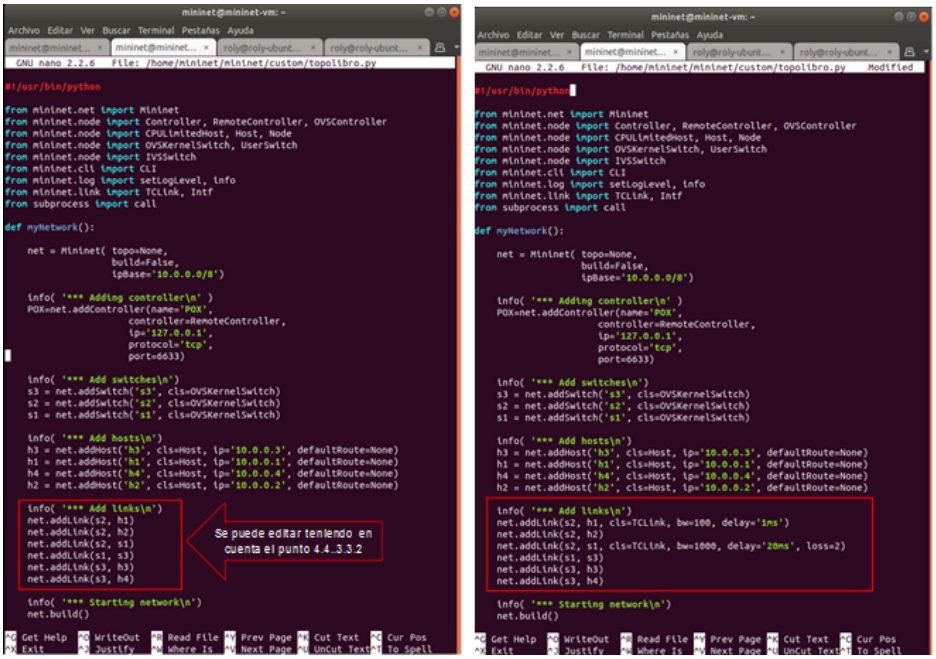

Figura 4.67: Ejemplo de script editada para agregar parámetros a los enlaces

La Figura 4.67, en el recuadro rojo, muestra un ejemplo donde al enlace entre el host 1 y el switch s2, se agregan parámetros de: ancho de banda 100 Mbps y un retardo de 1 ms. y al enlace entre los switch s2 y s1, se agregan parámetros de: ancho de banda 1000 Mbps, un retardo de 20 ms. y un porcentaje de perdidas 2 %. Siguiendo el ejemplo mostrado, se pueden agregar arbitrariamente los parámetros de enlace descritos, en el punto 4.4.3.3.2.

4.6.2.7. Menú RUN en MiniEdit

El menú RUN contiene los comandos de arranque / parada de la simulación, que cumplen la misma función de los botones de la parte inferior izquierda de la pantalla de la interfaz gráfica de usuario, y además algo muy interesante es que permite ver, mediante la opción RUN >Show OVS Summary (ver Figura 4.68).

La Figura 4.68, muestra el detalle de los puertos de los switches, también podemos verificar si el controlador está conectado y en que puerto está escuchando. Note que es exactamente lo mismo que ejecutar en linea de comando, como se expone en el punto 4.5.3.1 (Figura 4.70):

- *$ sudo ovs-vsctl show*

Comando sudo ovs-vsctl show. Para finalizar con el menú RUN, la opción RUN Root Terminal (Figura 4.70), nos entrega desde la interfaz gráfica una terminal de root de la MV Mininet, lo que resulta muy útil si se quiere utilizar cualquiera de los comandos de la familia OVS.

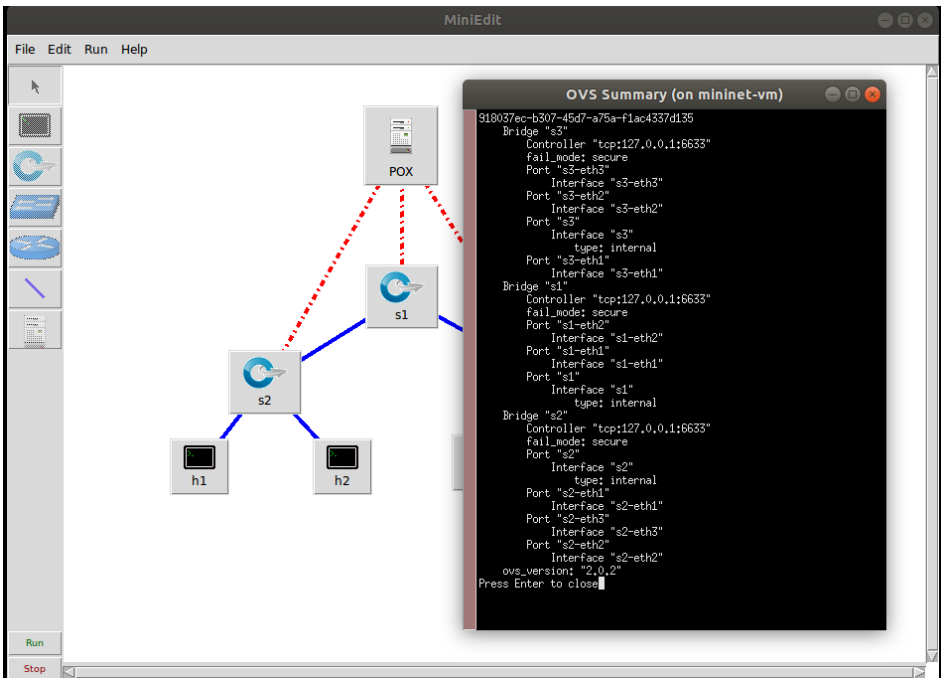

Figura 4.68: Salida de la opción RUN >Show OVS Summary en MiniEdit

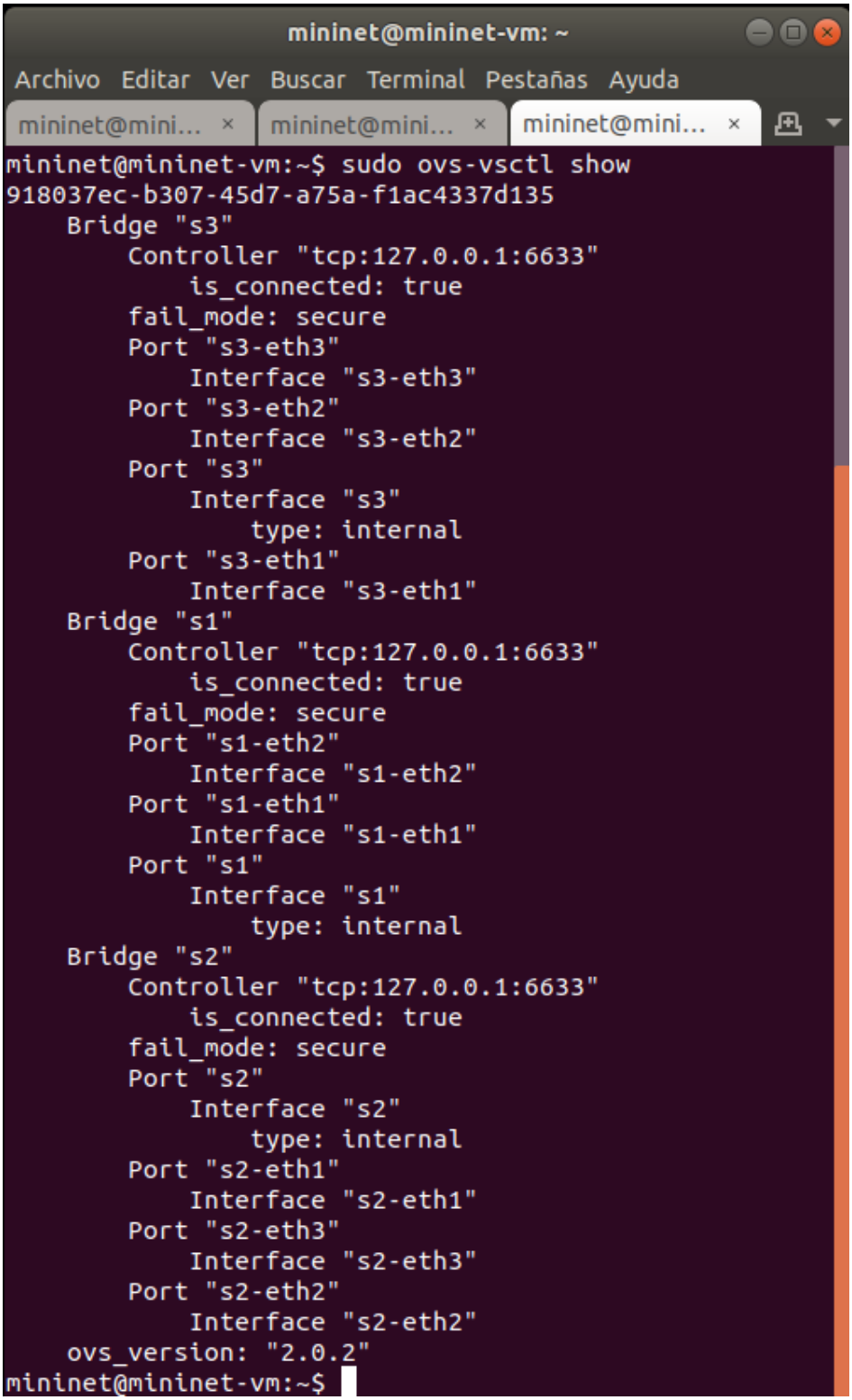

Figura 4.69: Comando sudo ovs-vsctl show

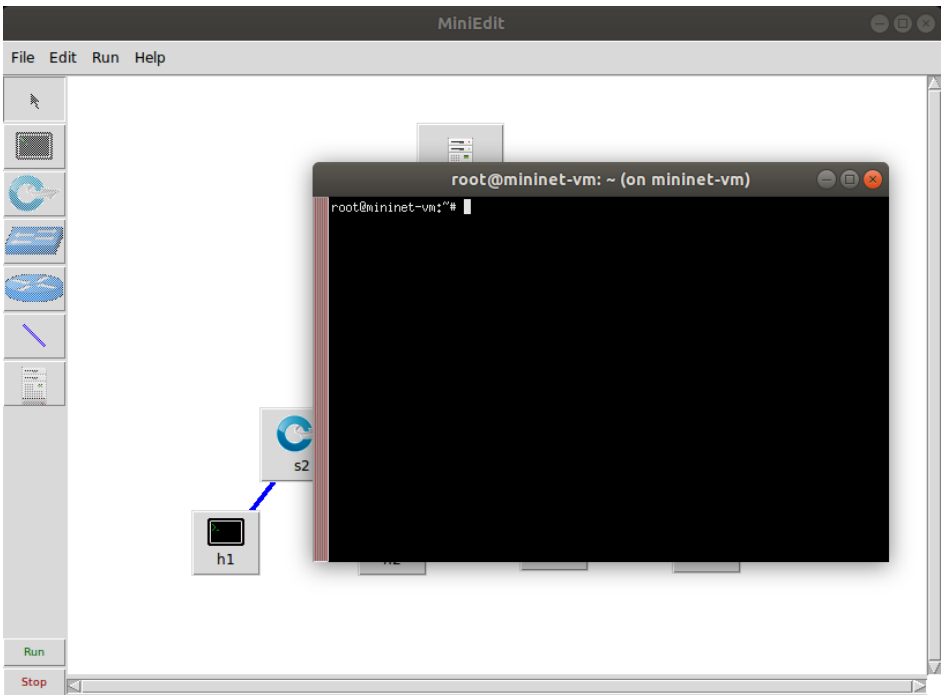

Figura 4.70: Menu RUN >Root Terminal de MiniEdit

4.7. Simulación con Mininet y Controlador OpenDaylight

En este punto se desarrolla la utilización del controlador OpenDaylight para la simulación y de muestran las configuraciones correspondientes.

4.7.1. Utilizando el controlador SDN OpenDaylight con el emulador de red Mininet

OpenDaylight (ODL) [73], (ver punto 3.1.3), en pocas palabras, es un framework de SDN open-source. Se utiliza para administrar una red emulada de switches y hosts virtuales emuladas con Mininet, en nuestro caso.

En este apartado se muestra la configuración de ODL para controlar una red emulada con Mininet utilizando OpenFlow 1.3. SE utilizan máquinas virtuales, pero el procedimiento es exactamente el mismo para un host con el sistema operativo: Linux, Windows y Mac OS X.

Se utilizan dos máquinas virtuales (VM). Una para ejecutar el emulador de red Mininet y la otra para ejecutar el controlador ODL. Se conectan ambas VMs como red host-only entonces van a poder comunicarse entre sí y con los programas corriendo en la computadora host, tales como ssh y X11 client.

Vamos a utilizar Virtual Box para correr la VM Mininet que podemos descargar desde la página oficial del proyecto [67]. El equipo de proyecto Mininet provee una imagen de VM Ubuntu 14.04 LTS con Mininet 2.2.1, Wireshark, OpenFlow y algunas otras herramientas ya instaladas y listas para usar. Si se quiere utilizar versiones más recientes de Linux, se recomienda descargar Mininet desde el repositorio del proyecto o también se instalar esta herramienta desde paquetes, ambas opciones se encuentran disponibles en la página del proyecto.

4.7.2. Construyendo la Máquina Virtual con OpenDaylight

Para crear una VM con ODL, descargar la Imagen de Ubuntu Server desde la página oficial de Ubuntu [96]. Luego se procede a instalar una nueva máquina virtual en VirtualBox.

Es conveniente poner un nombre descriptivo a la máquina virtual. En este caso podemos llamarla OpenDaylight. Vamos a configurarla para que utilice 2 CPUs y 2GB de RAM. Esta es la configuración mínima para soportar OpenDaylight. Luego vamos a configurar el adaptador de red de esta VM como host-only. Cuando la VM esta apagada, vamos a hacer click sobre el botón que dice Settings (Figura 4.71).

En la sección network de la VM (Figura 4.72), activamos dos interfaces de red. Conectaremos la primer interfaz de red al adaptador de red en modo Host-Only y el segundo adaptador de red en modo Bridge (aunque también se puede utilizar como NAT, es para tener salida a Internet).

4.7.3. Configuración de las interfaces en la Máquina Virtual OpenDaylight

Con la configuración mostrada en el punto anterior, se tienen dos interfaces disponibles en la VM, tal como muestra la Figura 4.73.

La interfaz eth1 tiene como Gateway al host anfitrión (que recomendamos tenga una IP estática).

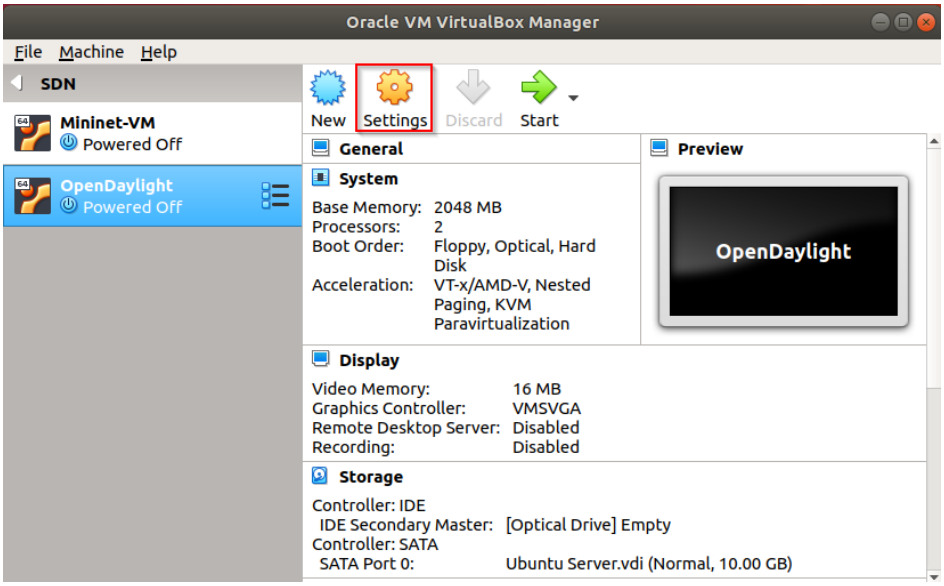

Figura 4.71: Configuración general de MV ODL

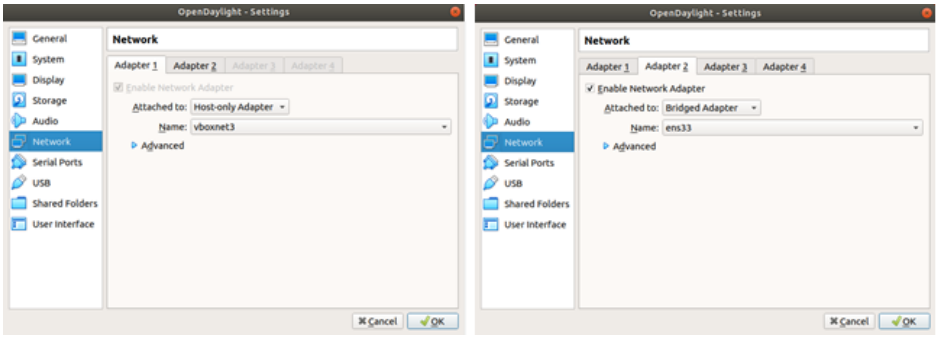

Figura 4.72: Configuracion de Red de MV ODL

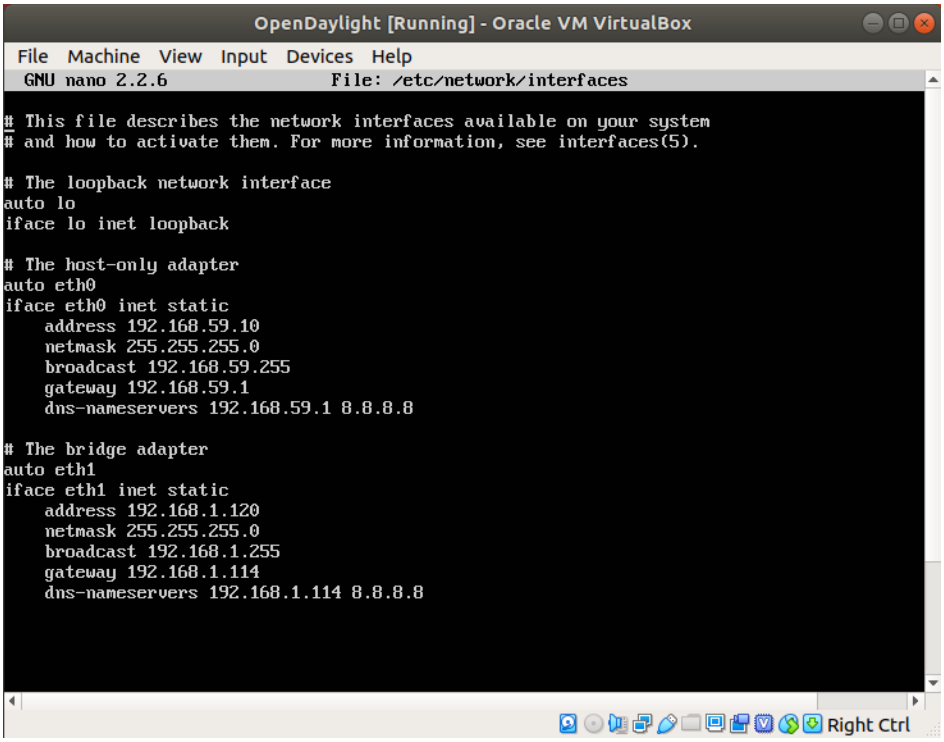

Figura 4.73: Configuración de las interfaces en la MV ODL

4.7.4. Configuración de la Máquina Virtual OpenDaylight vía SSH

Se accede a la VM ODL desde el host anfitrión vía SSH por medio del comando $ sudo ssh -X odl@192.168.59.10, donde odl es el nombre de usuario y la dirección IP es la que previamente configuramos como host-only en dicho servidor. Importante habilitar X11 forwarding a través de SSH con la opción X. De esta forma podemos ejecutar aplicaciones que tengan interfaz gráfica alojadas en el servidor en nuestro escritorio local. Para mayor información, se recomienda visitar la página oficial de Secure Shell [97].

4.7.5. Instalando Java en la MV OpenDaylight

El controlador SDN OpenDaylight es un programa escrito en Java. Podemos instalar el entorno de ejecución de Java con los siguientes comandos: • *$ sudo apt-get update* • *$ sudo apt install default-jre-headless*

***Nota:** se instala la versión 7 de java y no la última debido a que puede ocasionar errores de compatibilidad con la versión de ODL que vamos a utilizar. Las últimas versiones de ODL que se encuentran disponibles hasta la fecha utilizan java 8.*

***Nota:** Si se utiliza las versiones más actualizadas de Ubuntu Server como Ubuntu server 18.04 recordar que java 7 no se puede instalar directamente desde paquetes de instalación por el shell.*

***Nota:** Se puede instalar jdk y jre 7 de Ahora debemos establecer la variable de entorno JAVA_HOME. Esto lo vamos a realizar editando el archivo bashrc:*

- *$ sudo nano /.bashrc*

Añadiremos la siguiente línea al final de dicho archivo (Figura 4.74):

- *export* $JAVA_HOME = /usr/lib/jvm/default - java$

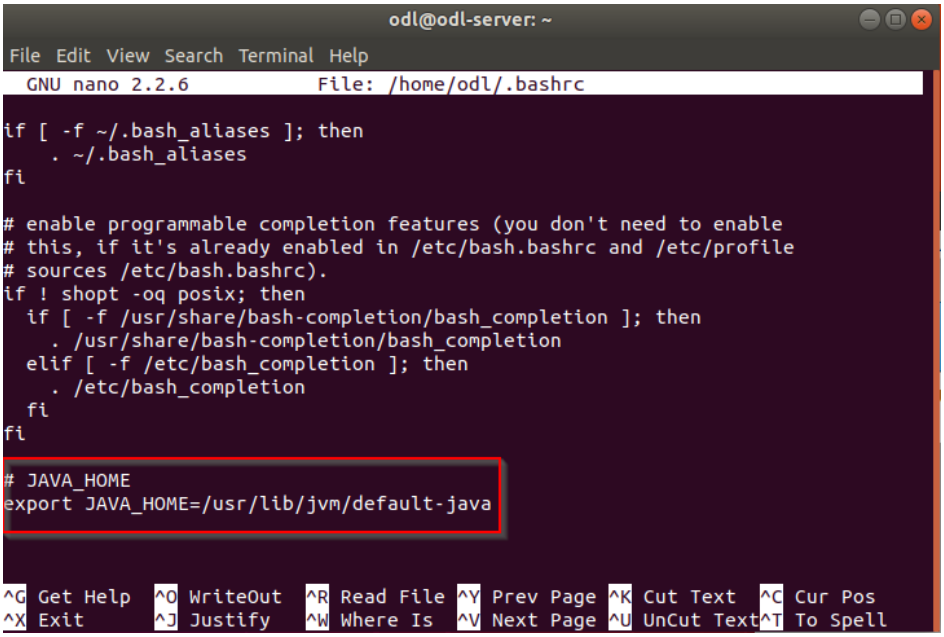

Figura 4.74: Editando el archivo bashrc

Luego de guardar, ejecutar el archivo: • *$ source /.bashrc*
Se verifica que se añadió correctamente la variable de entorno de la siguiente manera:

- *$ echo JAVA_HOME*

4.7.6. Instalando el software OpenDaylight en la MV

Se procede a descargar el software ODL desde la página web oficial de OpenDaylight [15]. En un host Linux o bien MacOS, podemos usar el comando wget para descargar el fichero de extensión tar. Para obtener el enlace de descarga debemos ir hasta la página oficial de ODL, ir hacia la sección de descargas:

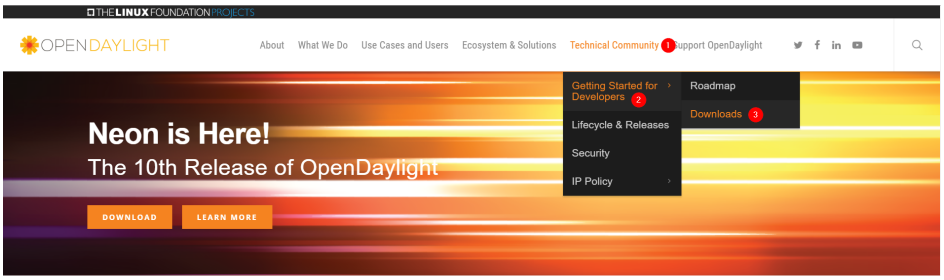

Figura 4.75: Página de descargas del proyecto OpenDayligh

Al navegar hacia la sección inferior de la página de descargas, encontraremos lo siguiente (Figura 4.76):

Al ser una versión antigua debemos ir hacia la sección indicada en la figura anterior. Una vez que accedamos debemos navegar hacia la última versión disponible de Beryllium (Figura 4.77):

Figura 4.76: Versiones archivadas del controlador OpenDaylight

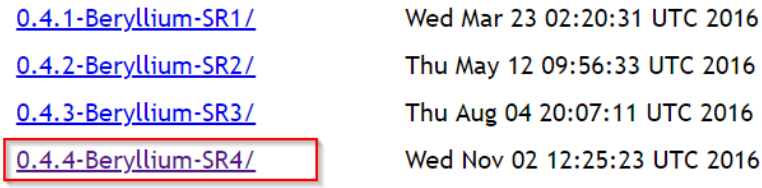

Figura 4.77: Listado de versiones anteriores del controlador OpenDaylight

Por último, podemos utilizar el click derecho en la página web para obtener el enlace de descarga del archivo de extensión tar (Figura 4.78):

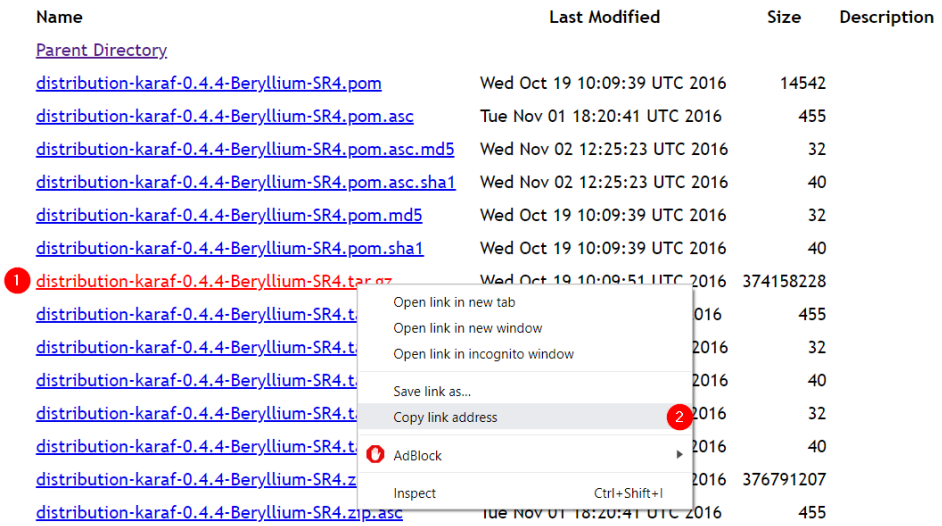

Figura 4.78: Seleccionando para descarga el archivo de controlador OpenDaylight

Cuando tengamos el enlace podemos copiarlo en la terminal, junto con el comando wget para comenzar la descarga: • $ wget https://nexus.opendaylight.org/content/repositories/public/org/opendaylight/integration/distributi karaf/0.4.4-Beryllium-SR4/distribution-karaf-0.4.4-Beryllium-SR4.tar.gz

**Nota: hay varias versiones de ODL, cada una con sus propias características, en nuestro caso instalamos la versión 0.4.4 llamada Belyllium, porque nos resulta más cómodo y práctico instalar la vm en esta máquina.*

Una vez descargado extraigamos el archivo tar:

- *$ tar -xvf distribution-karaf-0.4.4-Beryllium-SR4.tar.gz*

Esto creará un directorio llamado distribution-karaf-0.4.4-Beryllium-SR4 que contiene el software ODL y también los plugins.

ODL está empaquetado dentro de un contenedor Karaf. Donde Karaf es una tecnología de contenedor que permite a los desarrolladores colocar todos los requerimientos de software en una carpeta de distribución sencilla. Esto hace que sea muy sencillo de instalar o re-instalar ODL cuando sea necesario debido a que todo se encuentra en una carpeta. Como veremos más adelante Karaf también permite que los programas se agrupen en módulos opcionales que se pueden instalar cuando se necesite.

4.7.7. Ejecutando OpenDaylight

Para ejecutar ODL, utilizaremos los siguientes comandos: - *$ ./distribution-karaf-0.4.4-Beryllium-SR4/bin/karaf La salida se puede observar en la Figura 4.79.*

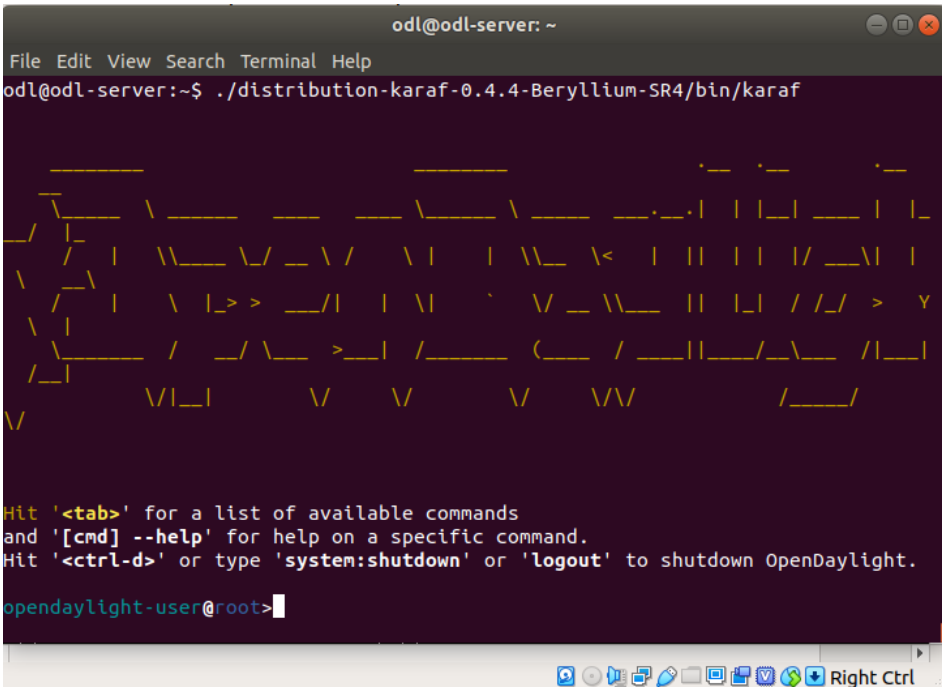

Figura 4.79: Ejecutando OpenDayligtht

4.7.8. Instalando las características de OpenDaylight

A continuación, vamos a instalar un conjunto de características mínimas de ODL y ODL GUI: En la terminal se escribe lo siguiente (Figura 4.80):

- *feature:install odl-restconf odl-l2switch-switch odl-mdsal-apidocs odl-dlux-all odl-openflowplugin-all*

Nota: Si por algun motivo se desea borrar las características instaladas se puede pasar como parámetro clean al ejecutar Karaf

- *$ ./distribution-karaf-0.4.4-Beryllium-SR4/bin/karaf clean*

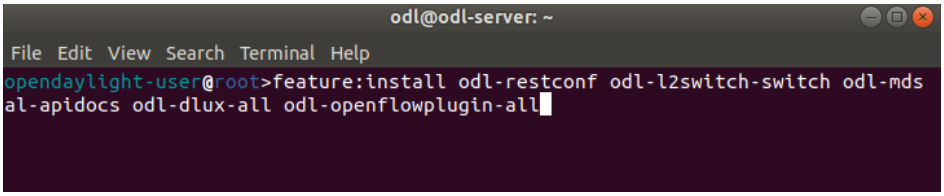

Figura 4.80: Instalando las características de OpenDaylight

Lo que hemos visto es un ejemplo de instalación de módulos opcionales en un contenedor Karaf. Sólo es necesario instalar las características opcionales una sola vez. Una vez instalada, esa característica se va añadir permanentemente al controlador y se inicializarán ni bien se ejecute Karaf. En esta ocasión instalamos las siguientes características:

- *odl-restconf: Permite el acceso a la RESTCONF API*
- *odl-l2switch-switch: Provee la funcionalidad de red similar a un switch Ethernet*
- *odl-mdsal-apidocs: Permite el acceso a la Yang API*
- *odl-dlux-all: Es la interfaz gráfica de ODL (funciona con java 7)*
- *odl-openflowplugin-all: Podemos obtener una lista de características opcionales ejecutando el comando:*

• *>feature:list La lista de las características instaladas se puede obtener ejecutando el comando:* • *>feature:list –installed*

La información sobre las características opcionales está disponible en la wiki de OpenDaylight.

4.7.9. Comprobando el funcionamiento del servidor

Nuevamente nos conectamos vía SSH al servidor ODL: • *$ sudo ssh -X odl@192.168.59.10*

Si instalamos la herramienta nmap [98], podemos observar los puertos abiertos de nuestra máquina con el comando:

• *$nmap localhost*

Como podemos observaren la Figura 4.81 nuestro servidor ODL esta escuchando desde el puerto 8181, que es donde podemos acceder a la interfaz gráfica desde el navegador.

4.7.10. La interfaz DLUX

La interfaz DLUX es una interfaz gráfica de ODL, funciona desde el navegador en el puerto 8181. Si accedemos a 192.168.59.10:8181/index.html desde nuestro navegador, donde 192.168.59.10 es la dirección IP de nuestro servidor ODL en nuestra red local. El usuario y la clave por defecto es admin. deberíamos ver lo que muestra la Figura 4.82.

***Nota:** Si la interfaz no carga o existe algún otro inconveniente debemos comprobar la versión de java y la variable de entorno* $JAVA_HOME$, *estainterfazfuncionaconjava*7, *versionessuperioresdeODLutilizanjava*8*paralai*

Para detener el controlador ODL, debemos ingresar la combinación de teclas <ctrl + d>o podemos escribir: system:shutdown o logout en el prompt del usuario. Tal y como se muestra cuando se inicia el controlador

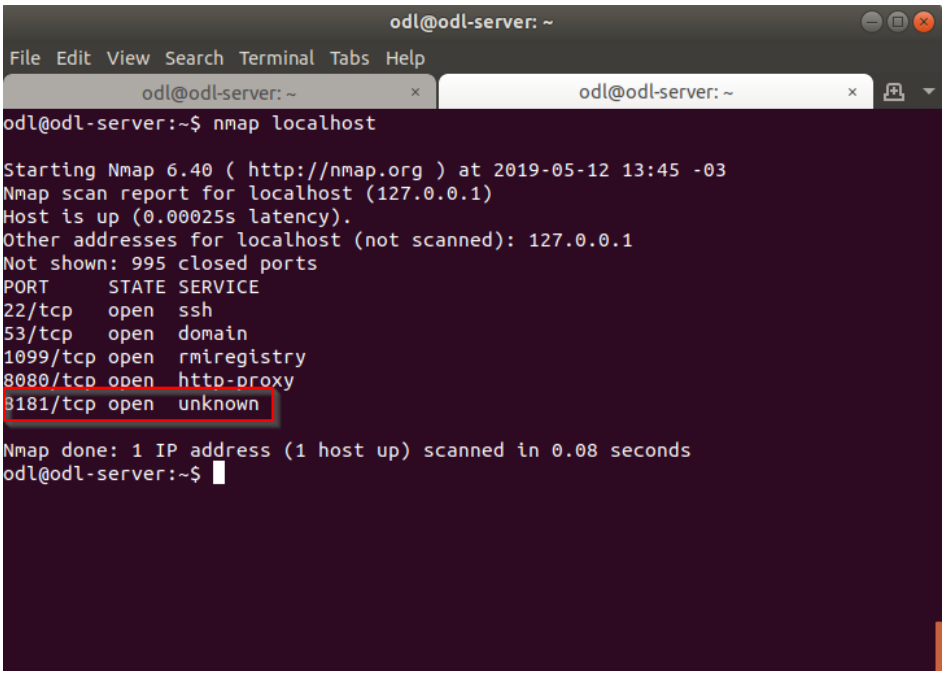

Figura 4.81: Comprobando servidor OpenDaylight con NMAP

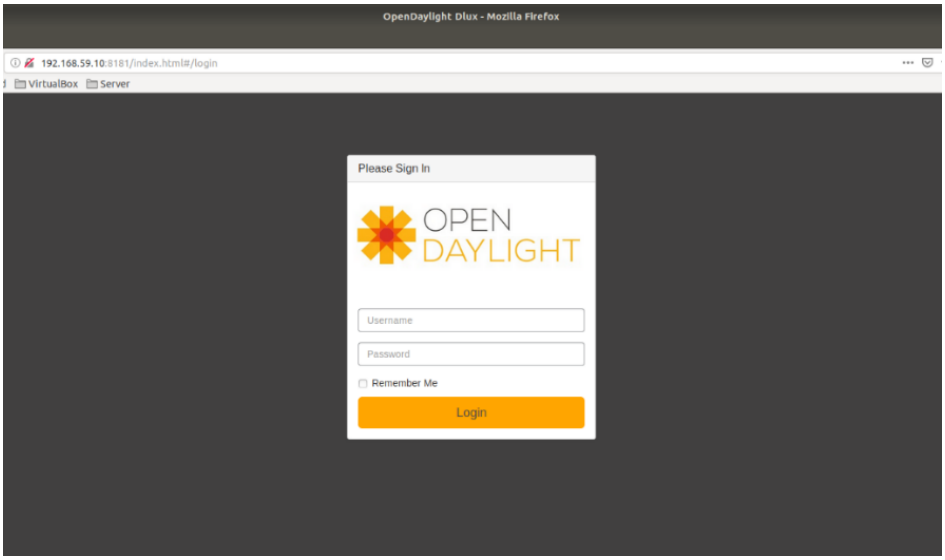

Figura 4.82: Interfaz Web DLUX

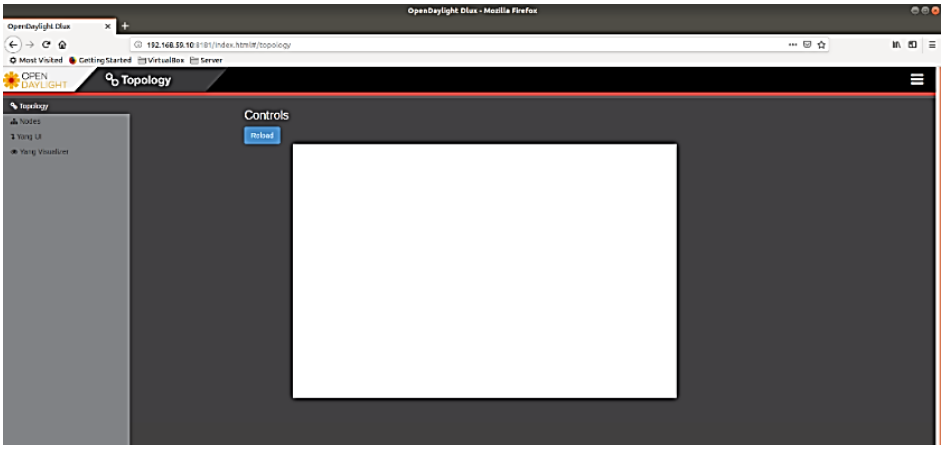

Figura 4.83: Captura de pantalla Dlux sección Topología

4.7.11. Configurando la Máquina Virtual Mininet para utilizarla con ODL

Teniendo la misma configuración que en el caso del Servidor ODL podremos editar la VM con Mininet. En este caso utilizaremos Ubuntu 18.04. Editaremos el un archivo de extensión YAML con el nombre XX-cloud-init.yaml, donde XX normalmente es un número y puede variar. Este archivo contine la configuración de red de Ubuntu 18.04 con el siguiente comando:

$ sudo /etc/netplan/50-cloud-init.yaml.

El nombre del archivo yaml pude variar solo asegurémonos que este archivo existe, en caso de que no lo podemos crear. Una vez allí según la información que obtuvimos con los comandos anteriores podemos editarlo para que tenga IPs estáticas como muestra la Figura 4.84:

Es de notar que ambas están en la misma subred.

4.7.12. Conexión con la Máquina Virtual Mininet vía SSH

Como en el caso del controlador ODL podemos conectarnos a la máquina virtual Mininet de la siguiente manera:

- *$ sudo ssh -X mininet@192.168.59.2*

**Nota:Si apagamos el servidor ODL debemos volverlo a activar. Recordemos que el controlador ODL se encuentra en la misma red en con la IP 192.168.59.10. Otra cuestión importante es que si en Mininet teníamos alguna topología debemos ejecutar el comando: sudo mn -c para eliminar cualquier configuración previa, así no estorba en nuestro experimento. Para establecer una topología se ejecuta el siguiente comando desde Mininet Figura 4.85:*

- *$ sudo mn –topo=linear,3 –mac –controller=remote,ip=192.168.59.10 –switch ovs,protocols=OpenFlow13*

4.7.13. Sección Topología de la Interfaz Gráfica OpenDaylight

Si accedemos a la interfaz gráfica ODL, llamada DLUX ahora deberíamos ver lo que muestra la Figura 4.86.

***Nota:** es posible que se requiera refrescar la página. La primera sección que se nos muestra en la interfaz se llama Topology, aquí podremos ver la red que se está emulando con el emulador de red Mininet. Se pueden construir distintas topologías de red en Mininet, con diferentes atributos, y utilizar*

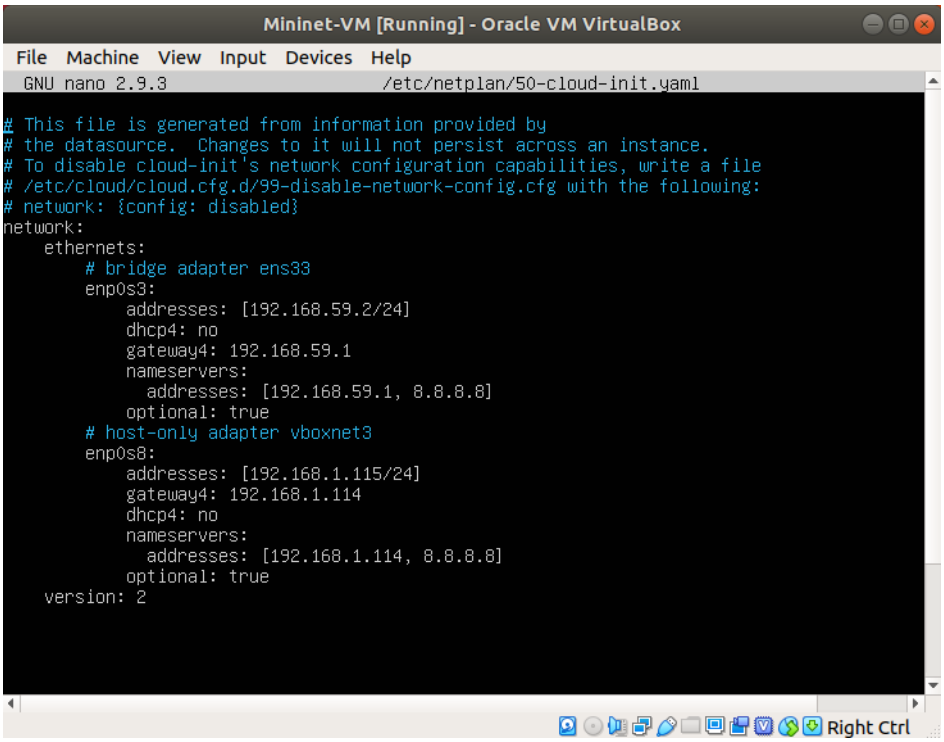

Figura 4.84: Configurando interfaces en la MV Mininet

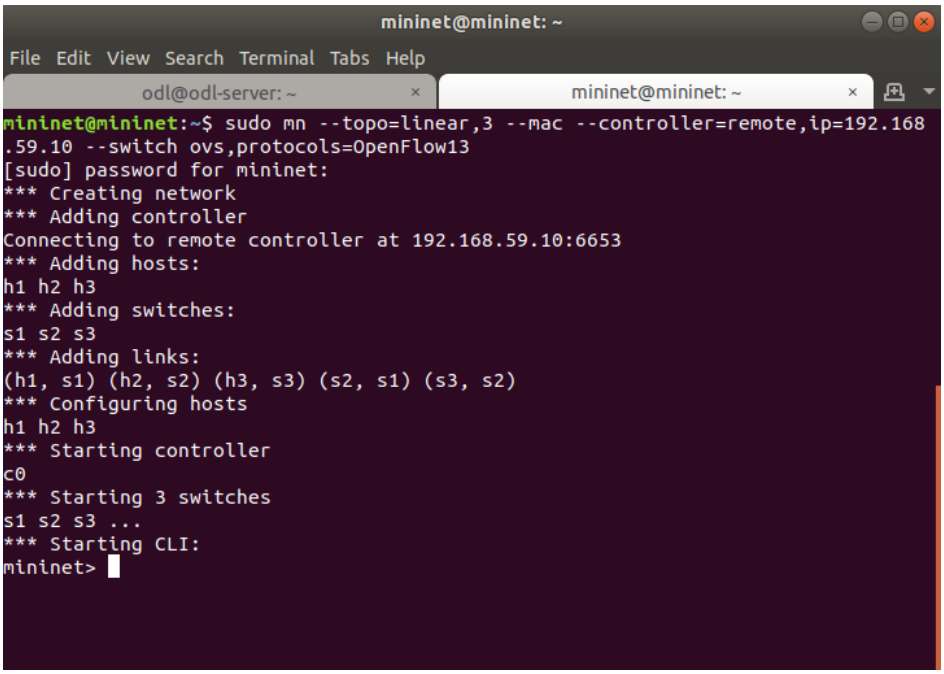

Figura 4.85: Arrancando Mininet con el controlador ODL

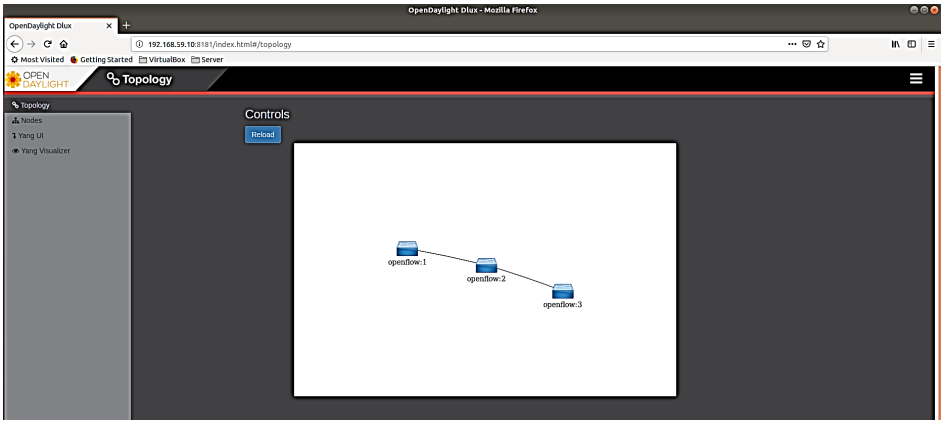

Figura 4.86: Topología de Mininet vista desde la interfaz gráfica del controlador ODL

ODL para realizar experimentos en ella. Por ejemplo, podríamos desconectar los switches para probar cómo se comporta la red ante fallas.

Es de notar que solo vemos los switches OpenFlow configurados, pero no los hosts, no nos debemos preocupar por ello. Si desde Mininet ejecutamos el comando pingall deberíamos ver lo siguiente Figura 4.87:

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet>
```

Figura 4.87: Ejecutando “pingall”desde minitet

Si ahora recargamos la interfaz web podremos observar lo que muestra la Figura 4.88.

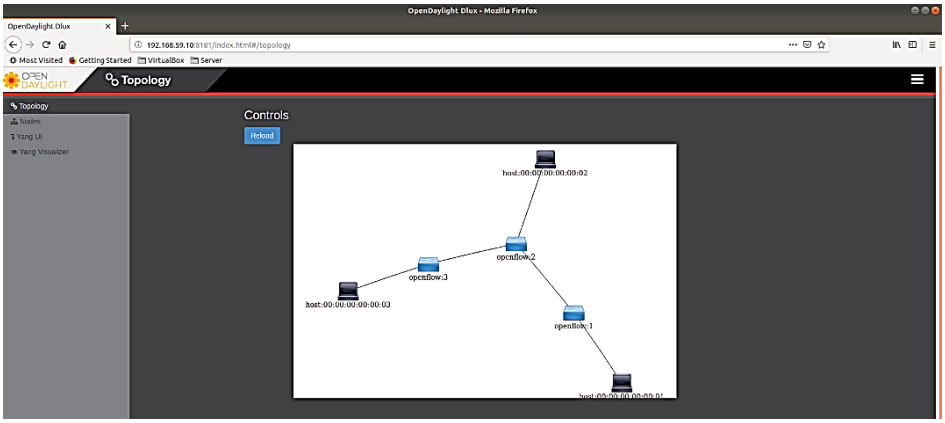

Figura 4.88: Topología de Mininet mostrada en la interfaz gráfica de ODL

4.7.14. Nodos de la interfaz gráfica OpenDaylight

Si hacemos click en la sección Nodes podremos ver información acerca de cada switch en la red, Figura 4.89.

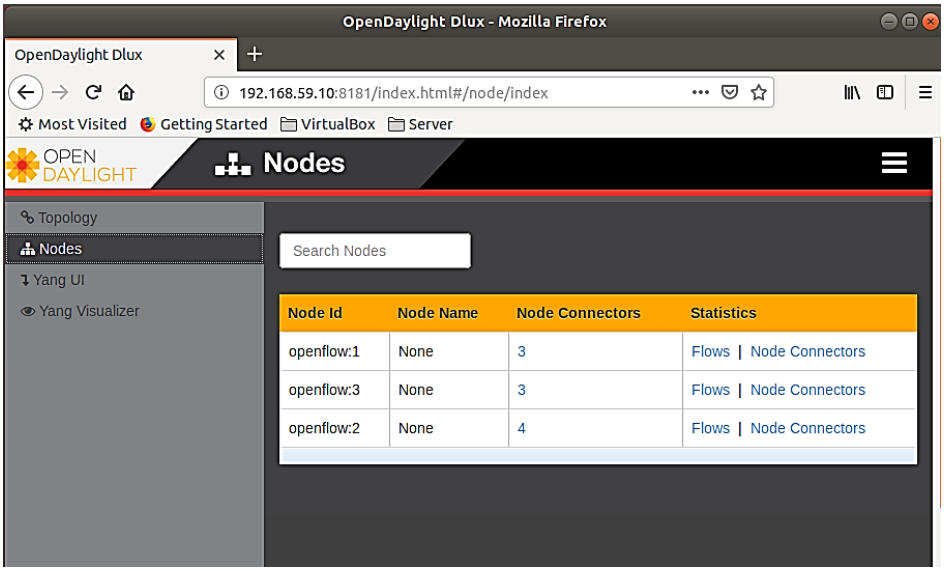

Figura 4.89: Sección Nodos interfaz gráfica ODL

Si hacemos click en algún conector de alguno de los nodos, podremos observar información detallada acerca de cada puerto perteneciente al switch Figura 4.90.

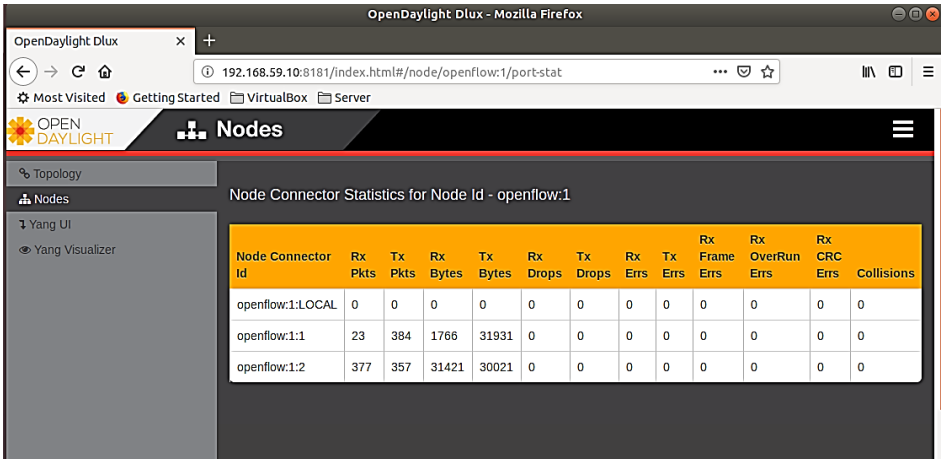

Figura 4.90: Captura de pantalla con información de los nodos de la interfaz gráfica ODL

4.7.15. Sección Yang UI de la interfaz gráfica de OpenDaylight

Yang es una estructura de modelado de datos. Los ingenieros que trabajan con routers de hardware y switches estarán familiarizados con otra estructura de datos de modelado basadas en SNMP, SMI y MIB. Yang provee una funcionalidad en switches SDN que es análogo a SMI para switches que no son SDN.

La interfaz de usuario ODL Yang es un cliente REST para crear y enviar REST request al almacen de datos ODL. Podemos utilizar Yang UI para obtener información desde el almacen de datos o crear comandos REST para modificar la información del almacen de datos -cambiando las configuraciones de red. Si hacemos click en la sección Yang UI y lugo hacemos click en botón Expand All podremos ver

todas las APIs disponibles, no es necesario que lo hagamos ahora. Además, no todas van a funcionar porque no hemos instalado todas las características. Una de las API que va a funcionar es la API de inventario llamada opendaylight-inventory. Si hacemos click en ella y luego en operational veremos el atributo nodes. Si hacemos click en el botón Send vamos a enviar el método GET API al controlador (Figura 4.91.

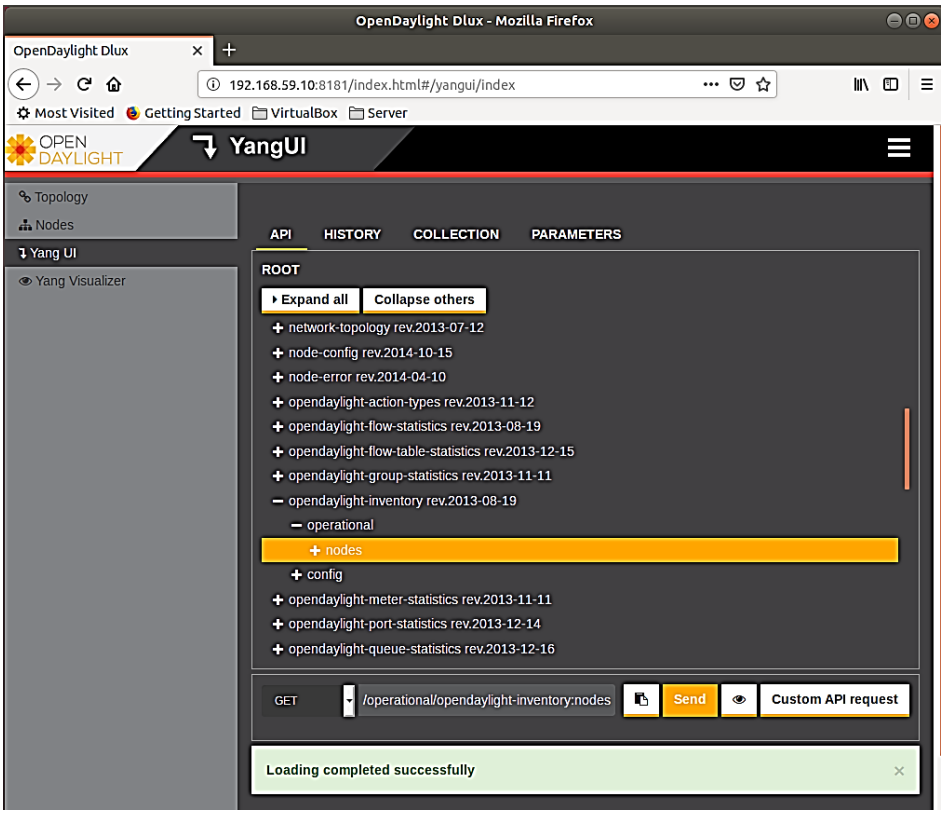

Figura 4.91: Sección Yang UI de la interfaz gráfica de ODL

Si enviamos y luego navegamos hacia abajo en la interfaz veremos toda la información del inventario, incluyendo: nodos, puertos, estadísticas, etc. Si hacemos click en los Switches obtendremos más detalles sobre los mismos Figura 4.92.

Comprender el modelo de datos Yang y aprender como leer y escribir datos en el almacén de datos es la clave para entender SDN con el controlador ODL Figura 4.93.

4.7.16. Capturando los mensajes OpenFlow

Para tener un aprendizaje más profundo sobre la operación de los controladores y los switches de SDN, podemos ver el intercambio de mensajes OpenFlow entre el controlador y los switches montados en la red. La VM Mininet también dispone de Wireshark. El camino más sencillo para visualizar los mensajes OpenFlow es ejecutar Wireshark en la VM donde tenemos Mininet y capturar los datos de la interfaz conectada a la red host-only, nuestra interfaz enp0s3 en este caso. Si abrimos otra sesión SSH en otra terminal hacia nuestra VM con Mininet con la opción X (o podríamos utilizar putty terminal por ejemplo si estamos en Windows):

- *$ sudo ssh -X mininet@ 192.168.59.2*

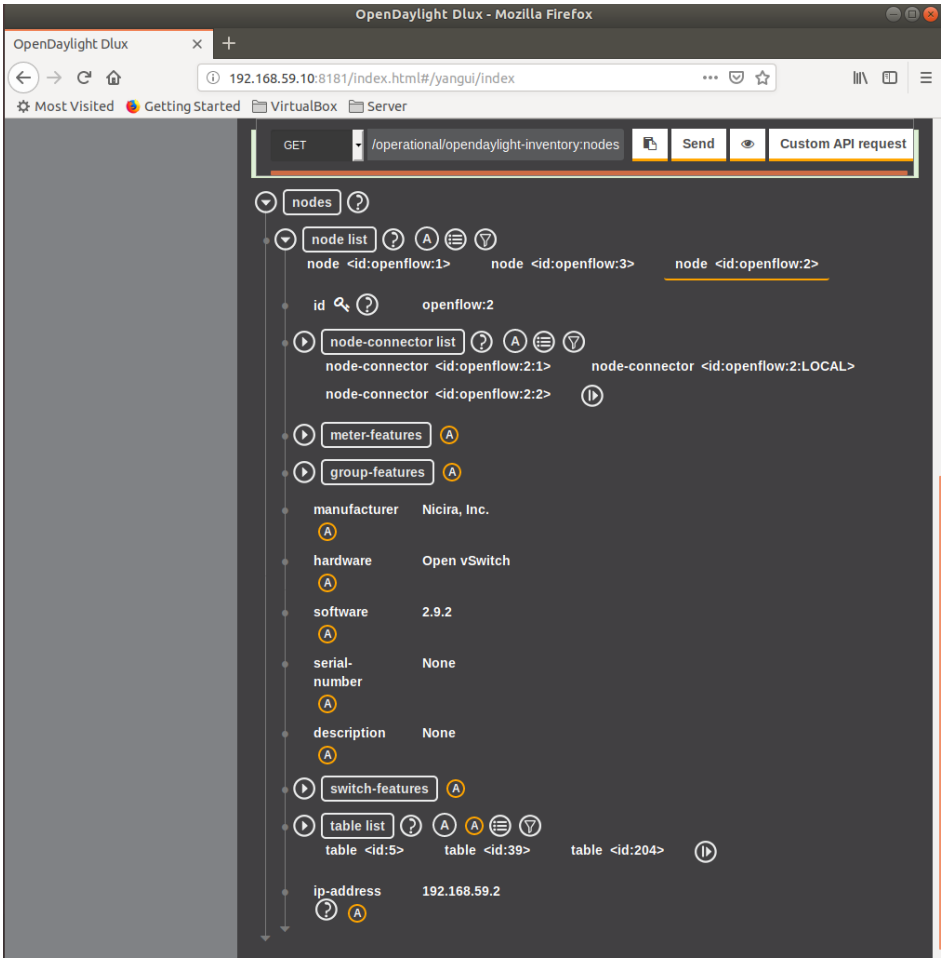

Figura 4.92: Obteniendo detalles y estadísticas de los switches mediante ïnventory-nodes

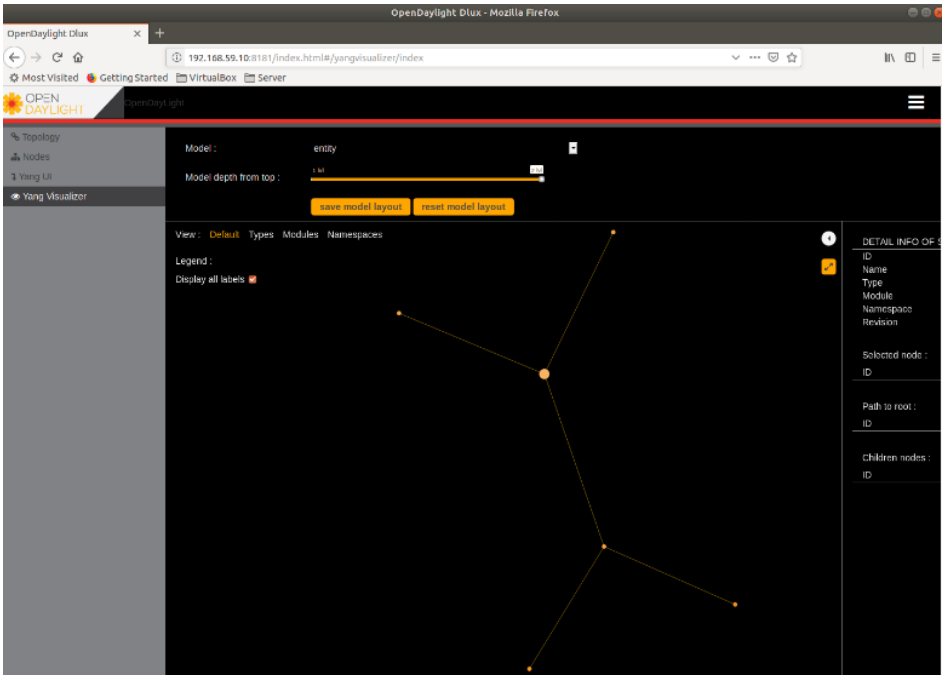

Figura 4.93: Modelo de datos Yang

Y luego ejecutamos wireshark en la máquina una vez conectados a ella, como muestra la Figura 4.94:

- *$ sudo wireshark &*

Muy probablemente nos aparezca un mensaje de dialogo diciéndonos que hubo un error, simplemente ignorémoslo y continuemos con nuestro laboratorio. Ejecutar Wireshark con privilegios root conlleva a riesgos de seguridad, pero por una simple prueba, nosotros podemos ignorar esto, también podríamos tener en cuenta el mensaje de advertencia para configurar Wireshark de una forma más segura. Ahora debemos de seleccionar la interfaz enp0s3 y hacemos doble click sobre ella.

Posteriormente aplicamos el filtro openflow_v4 (Figura 4.95)podremos ver los mensajes OpenFlow v1.3 que se envían.

4.7.17. Finalizando la prueba

Si queremos finalizar el proyecto, podemos apagar Mininet y ODL utilizando los siguientes comandos: En la VM de Mininet, debemos detener Mininet y limpiar el nodo, entonces para apagar la VM debemos ejecutar:

- *>exit*
- *>sudo mn -c*
- *$ sudo shutdown -h now*

En la VM donde corremos ODL ejecutamos:

- *>logout*

Para salir de Karaf. Y por último para apagar la VM:

- *$ sudo shutdown -h now*

Ahora VirtualBox cerrará las VM cuando ejecutemos los comandos anteriores. Desde el punto 4.7.4

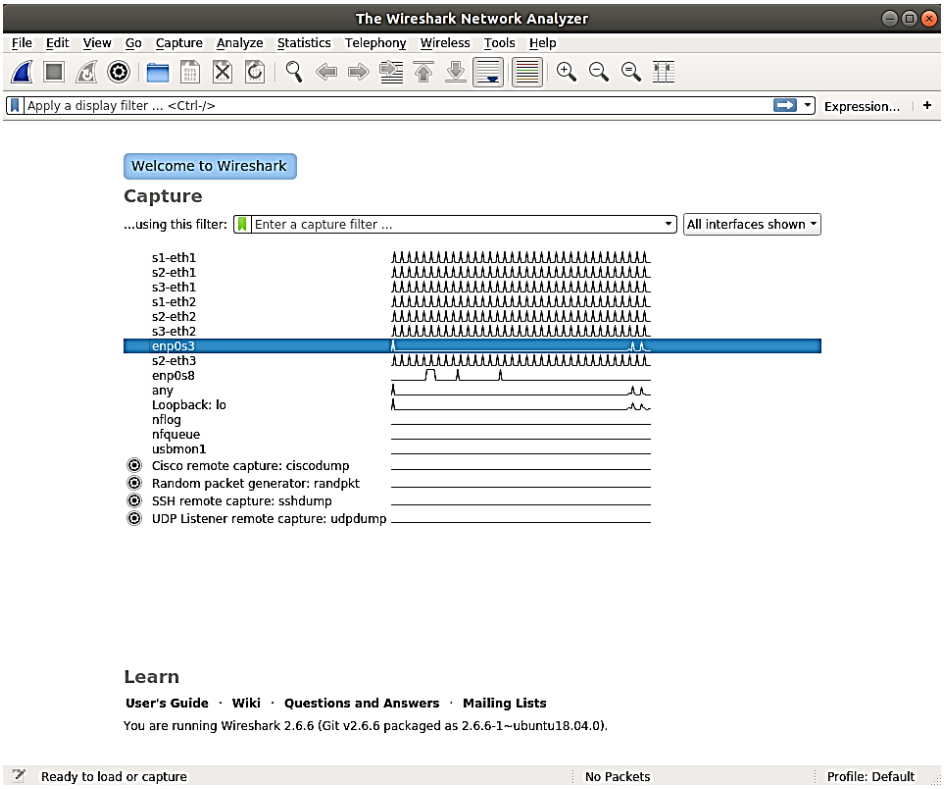

Figura 4.94: Captura de pantalla ejecutando Wireshark

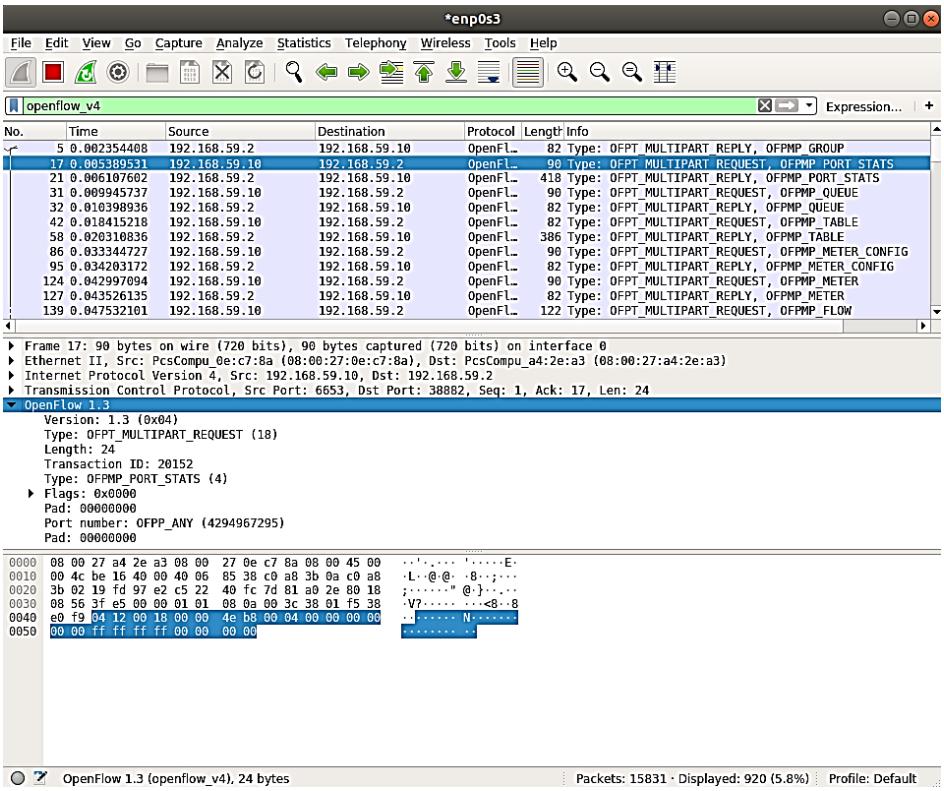

Figura 4.95: Wireshark con el filtro openflow$_v$4*aplicado*

al 4.7.16 hemos mostrado como instalar OpenDaylight en una máquina virtual y conectarla al emulador de red Mininet corriendo en otra Máquina virtual. Hemos demostrado algunas características de ODL y capturado algunos mensajes intercambiados entre el controlador los switches emulados.

4.8. Simulación con OpenDaylight versión Oxigen en Ubuntu Server 18.04

Si bien la configuración anterior es funcional, se utiliza Ubuntu 18.04 y Karn Oxigen (Karn 0.8.4) para esta sección, ya que son las versiones más actualizadas hasta la fecha y las que actualmente tienen mantenimiento.

4.8.1. Configuración de la Máquina Virtual Ubuntu Server 18.04 con OpenDaylight

Para crear una VM con ODL, como en el caso anterior, primero debemos descargar Ubuntu Server 18.04 desde la página oficial de Ubuntu. Debemos seguir los mismos procedimientos que en las secciones previas, en primer lugar, creamos una VM con 2 CPUs y 2GB de RAM. En segundo lugar, configuramos nuevamente dos adaptadores de red exactamente igual a lo que habíamos configurado para la VM con Ubuntu 14.04. Si lo hacemos de esta manera podemos tener ambos servidores en la misma red. Por último, para finalizar con este apartado, configuramos las interfaces de red dentro de la máquina virtual, como estamos trabajado en Ubuntu 18.04 esta configuración cambia con respecto a la configuración anterior, de la misma forma que en la máquina donde habíamos instalado Mininet debamos modificar el archivo de extensión yaml. Podemos hacerlo como muestra la Figura 4.96:

Conexión a la Máquina Virtual vía SSH Esto no difiere de los casos anteriores, según nuestra configuración anterior podemos conectarnos a nuestro servidor ODL utilizando el comando:

- *$ sudo ssh -X odl@192.168.59.4*

4.8.2. Instalando Java 8

Esta sección difiere de la configuración previa, debido a que las versiones actuales de ODL están escritos con la versión 8 de Java. Podemos instalar el entorno de ejecución de Java con los siguientes comandos:

- *$ sudo apt-get update*
- *$ sudo apt install oracle-java8-installer*

**Nota: Si bien es posible instalar versiones de ODL más antiguas en Ubuntu 18.04, es necesario utilizar Java 7 para este propósito y en esta versión de Ubuntu Server no se puede instalar dicha versión de java desde paquetes, lo cual hace un poco más engorrosa la instalación. Ahora debemos establecer la variable de entorno JAVA_HOME. Esto lo vamos a realizar editando el archivo bashrc (Figura 4.97):*

- *$ sudo nano /.bashrc*

Añadiremos la siguiente línea al final de dicho archivo ():

- *export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64*

Luego de guardar, ejecutamos el archivo:

- *$ source /.bashrc*

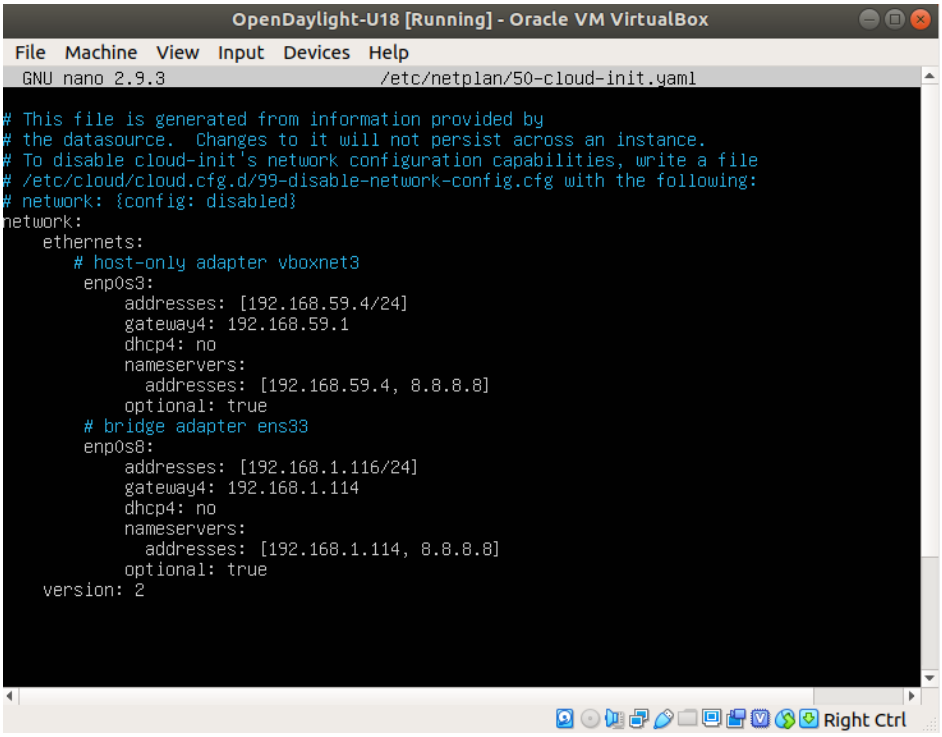

Figura 4.96: Configuración de las interfaces en Ubuntu 18.04

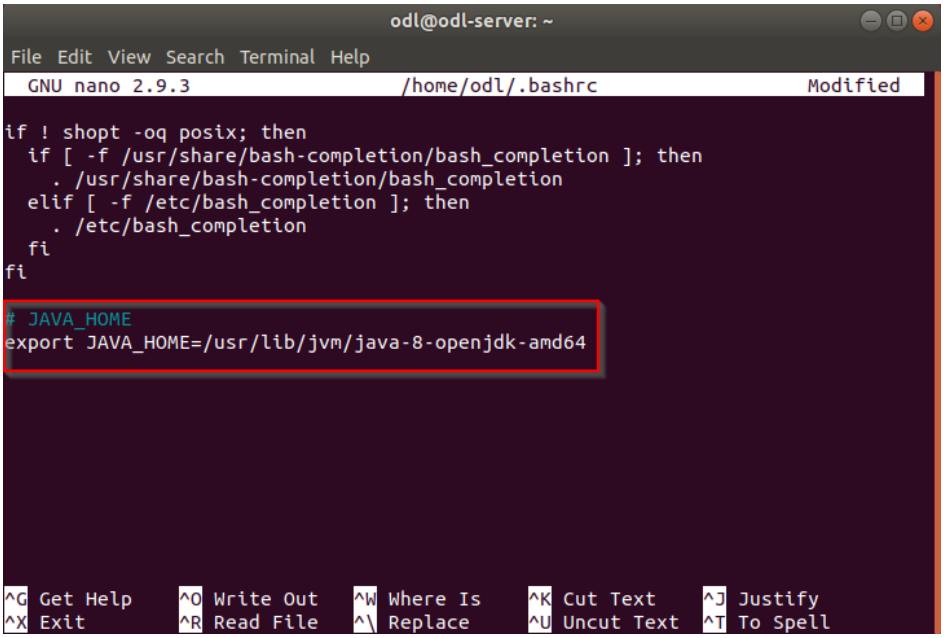

Figura 4.97: Editando el archivo bashrc para Java 8

4.8.3. Instalando OpenDaylight versión Oxigen

En esta ocasión vamos a instalar la versión Oxigen de ODL, que lo podemos descargar desde su página oficial.

https://docs.opendaylight.org/en/stable-Oxigen/downloads.html

Es la página principal de descargas donde se encuentran las últimas versiones y copiamos el enlace como muestra la Figura 4.98:

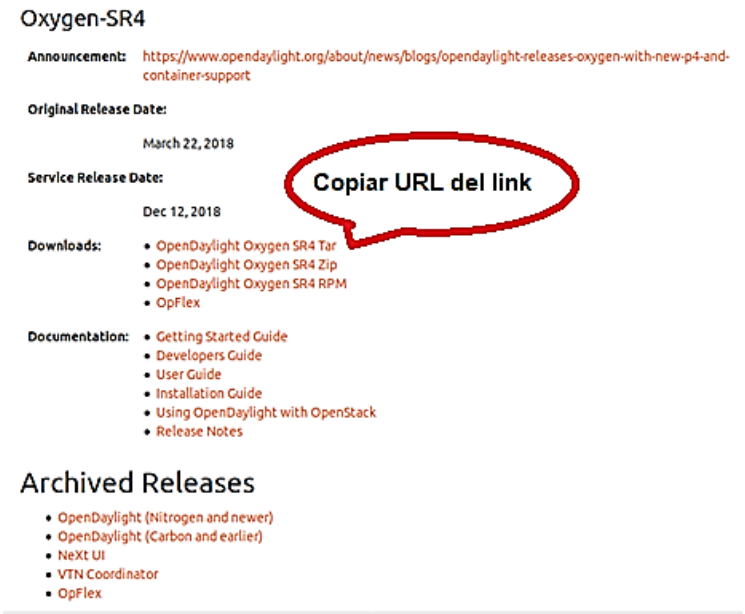

Figura 4.98: Copiar link para la versión Oxigen de ODL

Para descargarlo ejecutamos:

- *$ wget https://nexus.opendaylight.org/content/repositories/public/org/opendaylight/integration/karaf/0.8.4/karaf-0.8.4.tar.gz*

Para descomprimirlo:

- *$ tar -xvf karaf-0.8.4.tar.gz*

Ejecutando OpenDaylight Para ejecutar ODL, utilizaremos los siguientes comandos:

- *$ cd karaf-0.8.4*
- *$./bin/karaf*

La pantalla de la aplicación Karaf no difiere de las versiones anteriores, pero es muy importante destacar que para esta versión la mayoría de los comandos que utilizamos en la configuración más antigua de ODL se encuentran deprecados. En este caso para tener las mismas prestaciones, una vez que el servidor está activo, debemos ejecutar:

- *>feature:install odl-l2switch-switch odl-dlux-core odl-dluxapps-applications*

Tal como muestra la Figura 4.99.

Para comprobar que el servidor este corriendo podemos abrir una nueva terminal y conectarnos vía SSH nuevamente a esta máquina para ejecutar nmap como muestra la Figura 4.100:

- *$ nmap localhost*

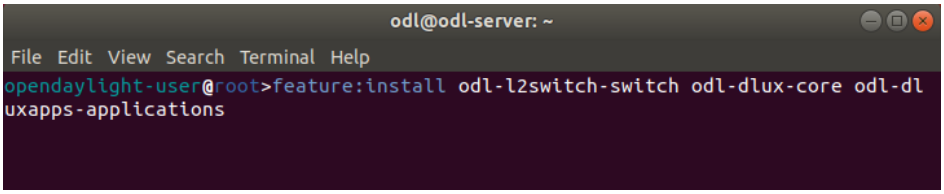

Figura 4.99: Instalando características en ODL versión Oxigen

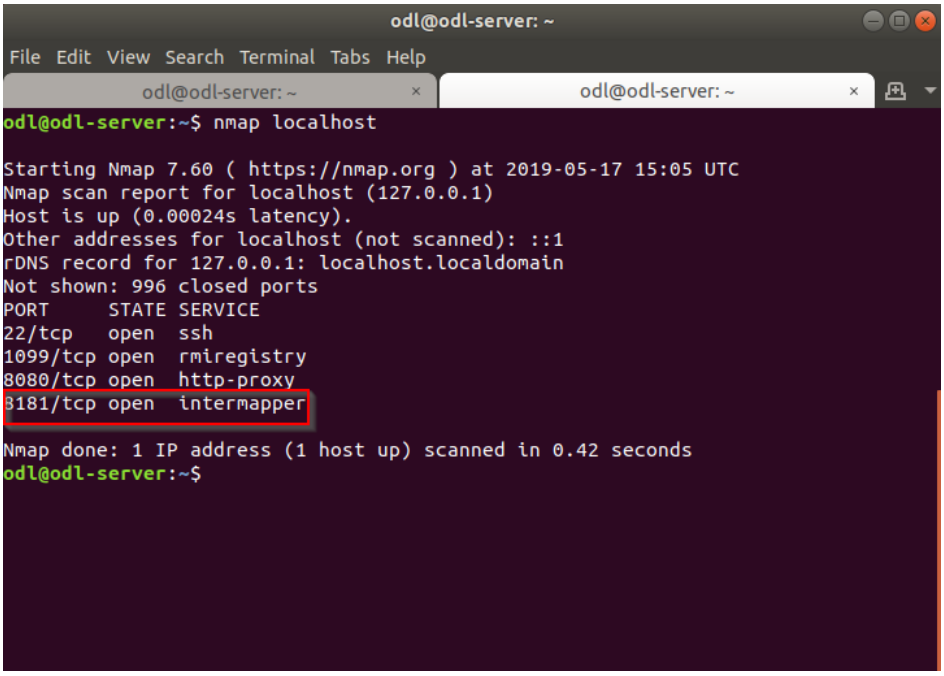

Figura 4.100: Comprobando el servidor ODL con NMAP

4.8.4. Ejecutando Mininet con ODL versión Oxigen

Ahora es necesario conectarnos a Mininet vía SSH y crear una nueva topología junto a este servidor ODL, de manera semejante a lo que habíamos hecho anteriormente. En este caso los comandos serían los siguientes:

- *$ sudo ssh -X mininet@192.168.59.2*

Es de notar que vamos a utilizar la misma VM Mininet configurada anteriormente. Cuando nos hayamos conectado a Mininet vamos a poder crear una nueva topología con el nuevo servidor ODL, no olvidemos utilizar el comando sudo mn -c para limpiar configuraciones anteriores y evitar tener problemas.

- *$ sudo mn –topo=linear,3 –mac –controller=remote,ip=192.168.59.4 –switch ovs,protocols=OpenFlow13*

Con este comando veremos la pantalla tal como muestra la Figura 4.101.

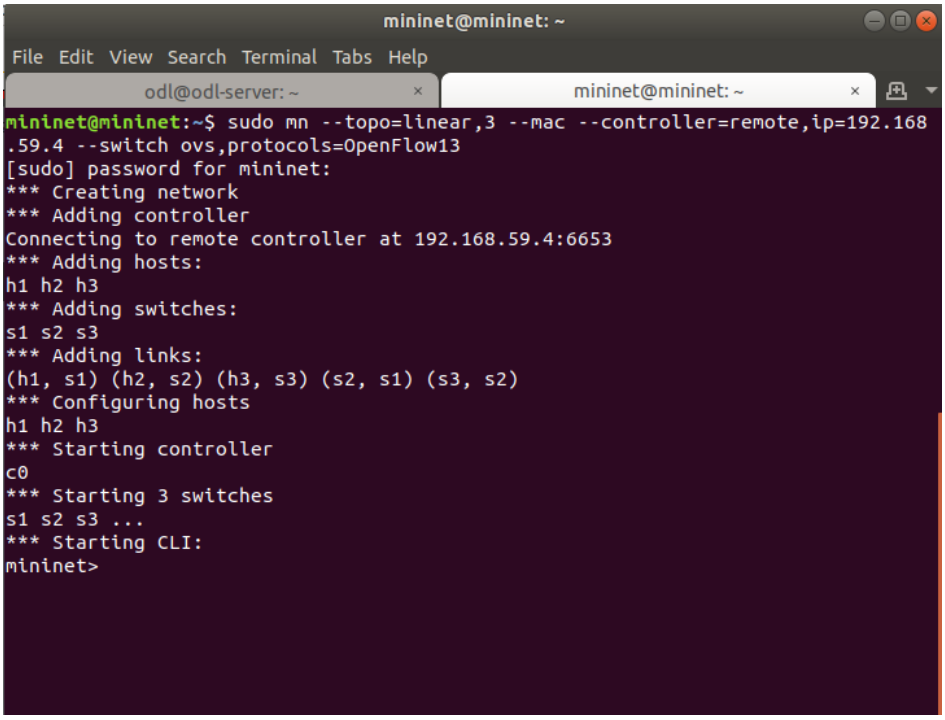

Figura 4.101: Topología creada en Mininet

Una vez creada la topología, de la misma forma que en el caso anterior, abrimos nuestro navegador y nos dirigimos a 192.168.59.4:8181/index.html, tal como muestra la Figura 4.102.

Luego en mininet ejecutamos el comando (Figura 4.103)

- *>pingall*

Y podremos obtener la topología que hemos creado (Figura 4.104):

Sección Yangman de la interfaz gráfica de OpenDaylight Oxigen Yangman es una aplicación basada en ODL DLUX. Es utilizado en el software estándar ODL incluyendo MD-SAL y RESTCONF. ODL genera dinámicamente REST APIs a partir de modelos o modulos Yang, lo que proporciona a los desarrolladores de aplicaciones ODL un conjunto de APIs de referencia. Es preciso que aclaremos que Yang es un lenguaje de modelado de datos (Figura 4.105).

Yangman utiliza mucho código que ya estaba presente en YangUI, de hecho, en el ejemplo de esta

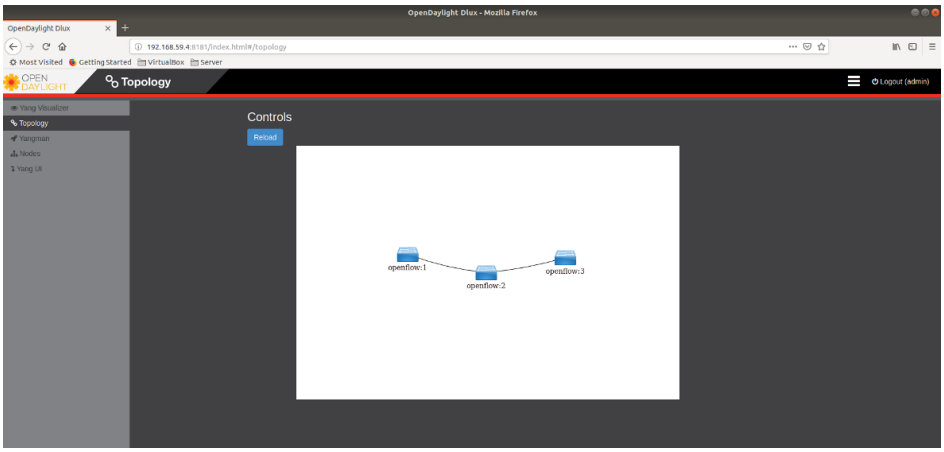

Figura 4.102: Topología de Mininet mostrada en la interfaz gráfica de ODL versión Oxigen

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet>
```

Figura 4.103: Pantalla de Comando pingall

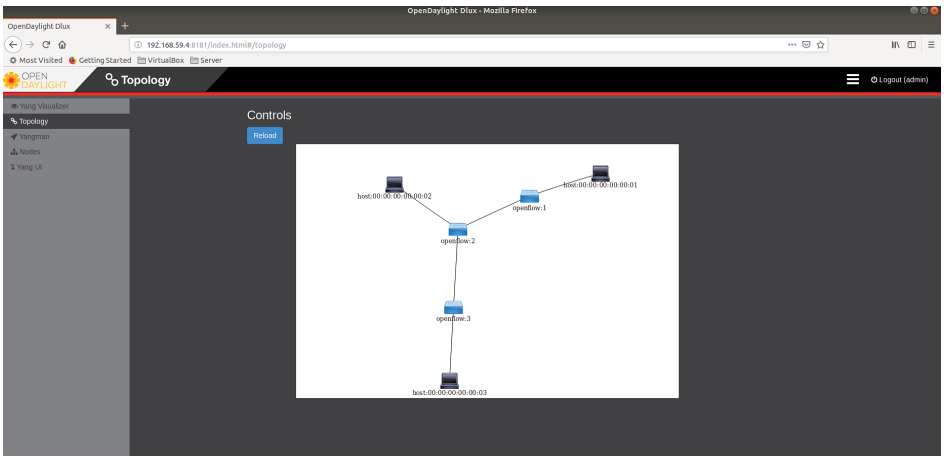

Figura 4.104: Topologia de Mininet mostrando los hosts luego del comando pingall

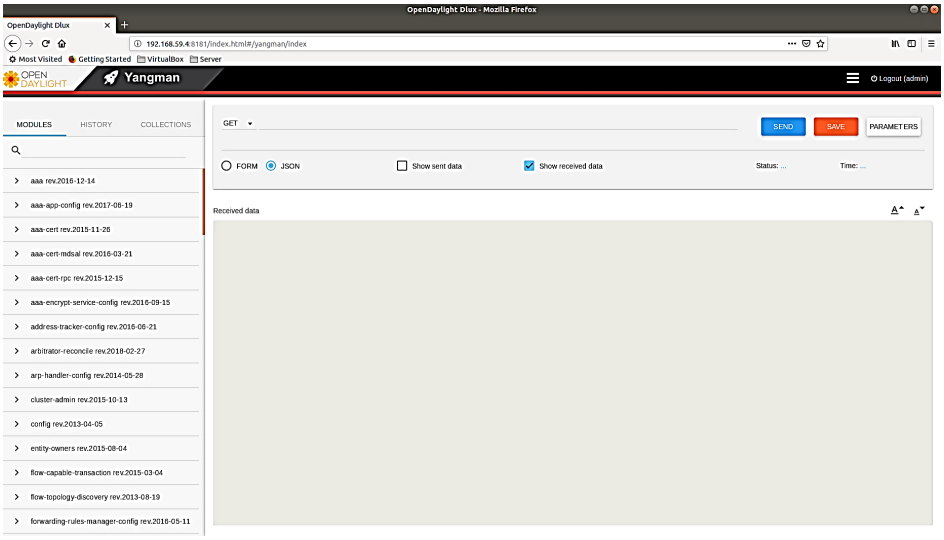

Figura 4.105: Yangman de ODL versión Oxigen

sección veremos varias similitudes. La principal diferencia es la interfaz gráfica de usuario, similar a Postman. Esta última aplicación es muy conocida en la industria para trabajar con REST APIs. Yangman GUI nos permite realizar operaciones en una interfaz web dentro del sistema. La GUI está diseñada e implementada en AngularJS, que comúnmente utiliza el patrón modelo-vista-controlador (MVC). Internamente el sistema define los modelos, los servicios que se aplican a esos modelos, los controles para manipular los datos en dichos modelos y, por último, las vistas (por ejemplo, una página html) que son desplegadas en un navegador y están vinculadas a un controlador. Este patrón de diseño MVC provee un entorno de desarrollo que puede ser reutilizado en varias aplicaciones. Por supuesto, esto está implementado dentro de ODL DLUX. Para más información recomendamos visitar la página del proyecto [73]. En este apartado daremos un breve vistazo a la interfaz de Yangman.

En la Figura 4.105 podemos ver una lista desplegable de los modulos Yang después de haber ejecutado Yangman. Si seleccionamos el algún modulo, luego tendremos una porción del árbol para trabajar. Como en apartados anteriores podemos navegar hacia abajo e ir a odl-inventory. En la parte izquierda veremos un panel que podremos mostrar/ocultar que muestra una visualización del módulo Yang seleccionado, que se utiliza con fines de referencia.

También tenemos un buscador encima donde podremos buscar por ejemplo opendaylight-inventory (Figura 4.106), que habíamos buscado en las configuraciones anteriores.

Al seleccionarlo y se desplegará una lista, que sólo contiene el elemento operacional en nuestro caso (Figura 4.107).

Si a continuación hacemos click allí veremos una lista de nodos, de la misma forma que nos mostraba YangUI.(Figura 4.108)

Cuando apretemos el botón de enviar (dejando seleccionado nodes en el panel izquierdo), se transmitirá al controlador ODL la REST API con el cuerpo especificado. Los formularios de la interfaz (autogenerados por Yangman) nos permitirá probar nuevas REST APIs, podemos seleccionar tanto formularios como JSON (Figura 4.109 y Figura 4.110).

La casilla seleccionable de JSON, que lo encontraremos en la esquina superior, nos permitirá cambiar a la pantalla a JSON (Figura 4.111).

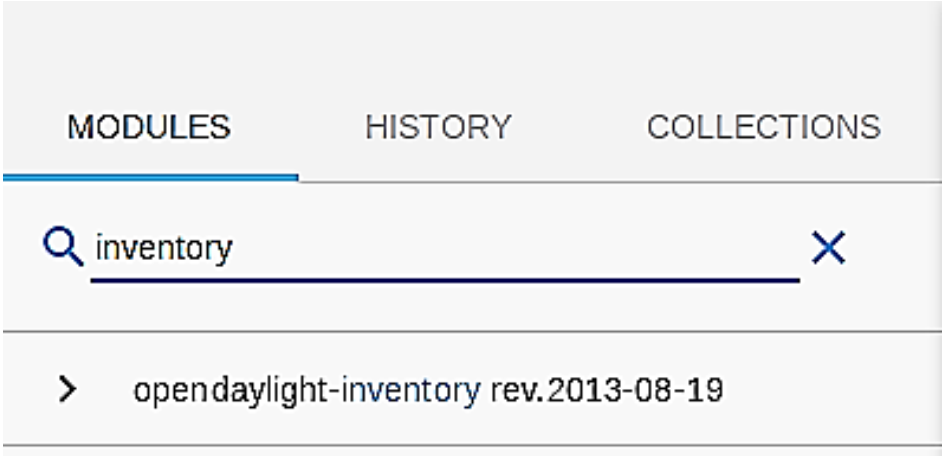

Figura 4.106: Modulo opendaylight-inventory

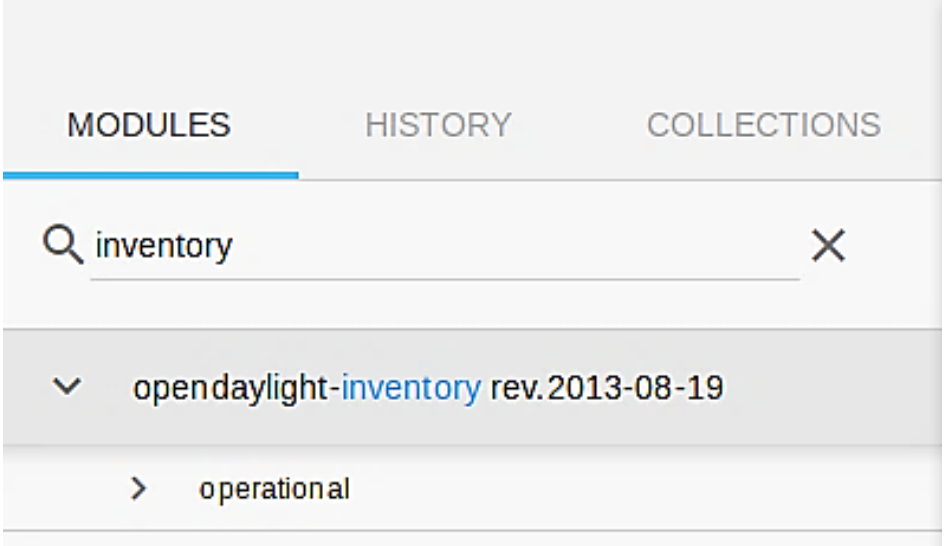

Figura 4.107: Elemento operacional

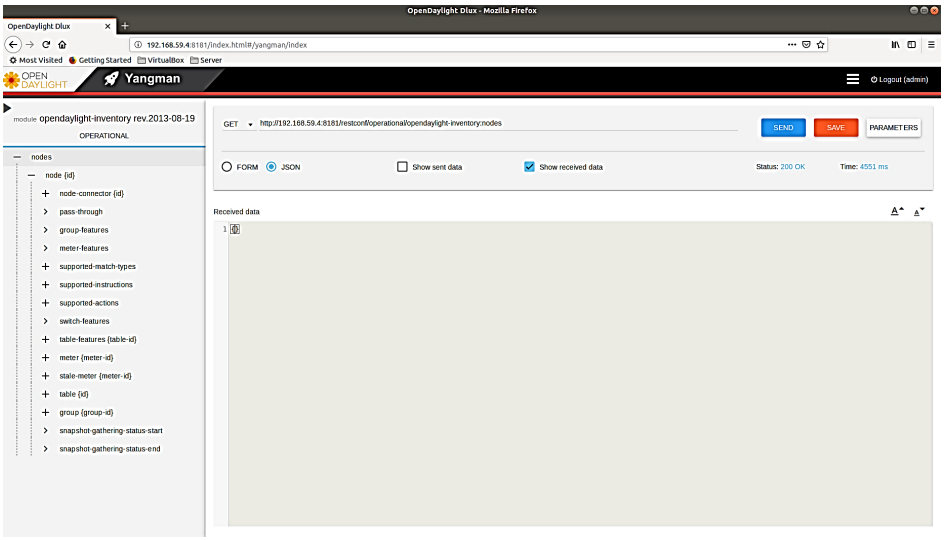

Figura 4.108: Yangman Inventory nodes

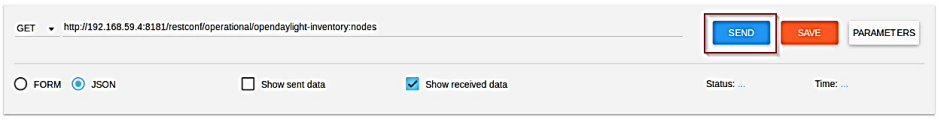

Figura 4.109: Transmisión de los formularios generados por Yangman

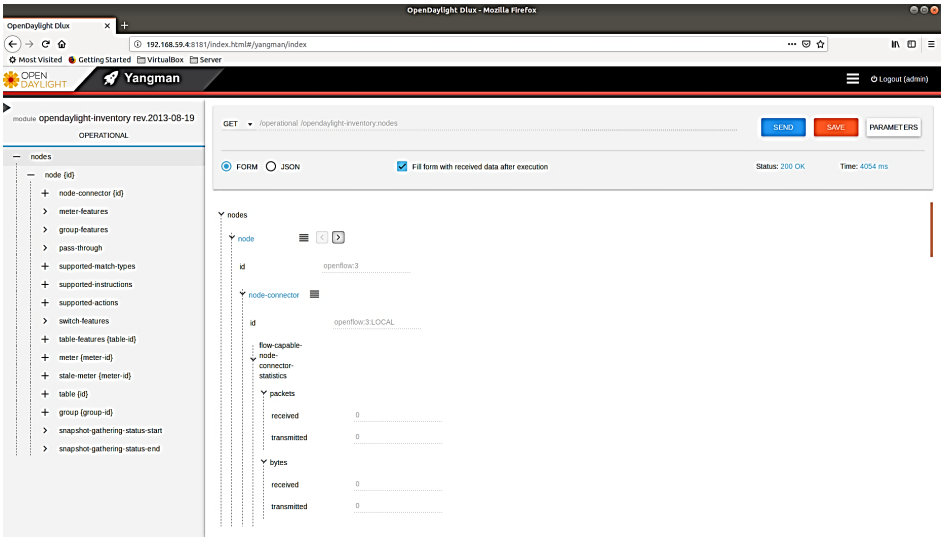

Figura 4.110: Formulario generado por Yangman

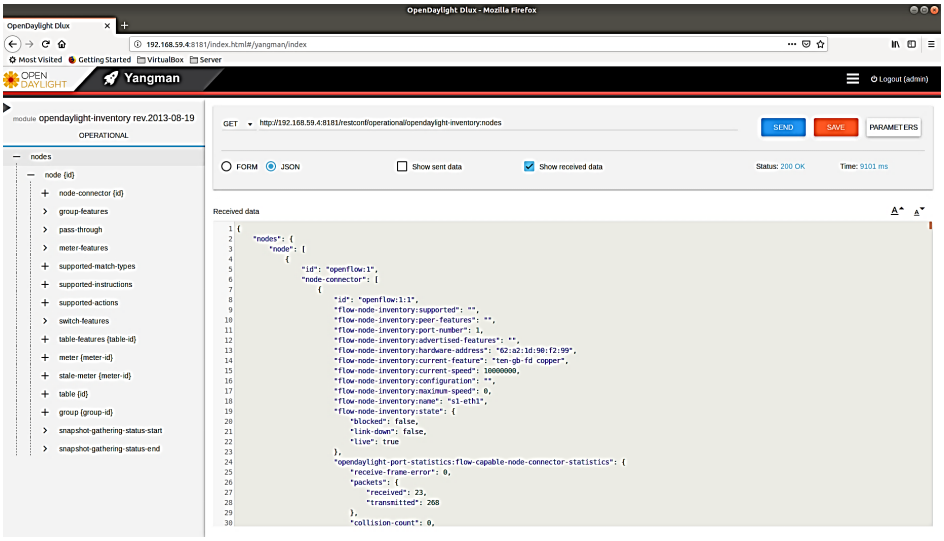

Figura 4.111: Datos en formato JSON

Si estamos familiarizados con Postman, nos daremos cuenta que el panel es muy similar. Es posible alternar entre el formato JSON y el formato de formulario. Una característica heredada de YangUI nos permitirá a los usuarios agregar, modificar o eliminar parámetros que pueden ser fácilmente editados en la pantalla de JSON.

Si volvemos a la sección anterior haciendo click en la esquina superior izquierda, veremos nuevamente el panel principal. (Figura 4.112)

Las siguientes figuras muestran el historial y colecciones respectivamente, nuevamente esto es heredado de Postman (Figura 4.113).

Si no tenemos ninguna colección guardada, dentro de la misma pestaña y con una respuesta a una petición previamente realizada (la que realizamos anteriormente con GET por ejemplo), podemos guardarla utilizando el botón de guardado, en donde luego nos pedirá un nombre significativo como muestra la Figura 4.114.

Una vez guardada la podremos ver en las colecciones (Figura 4.115).

Desde el punto 4.8 vimos algunas características de una versión más actualizada de ODL, con respecto a la sección previa. Pudimos notar que la diferencia más significativa está en la instalación de características al servidor ODL, ya que cambió algunos comandos con respecto a versiones anteriores. Por otra parte, la interfaz DLUX es semejante a la anterior salvo que se agrega la característica de Yangman que posee una interfaz muy amigable y aún más si estamos familiarizados con la aplicación Postman, mejorando y agregando más características de las que ya tenía YangmanUI.

- *url de referencia de Yangman: https://github.com/CiscoDevNet/yangman*
- *url pdf cisco live: https://www.ciscolive.com/c/dam/r/ciscolive/emea/docs/2018/pdf/DEVNET-1119.pdf*

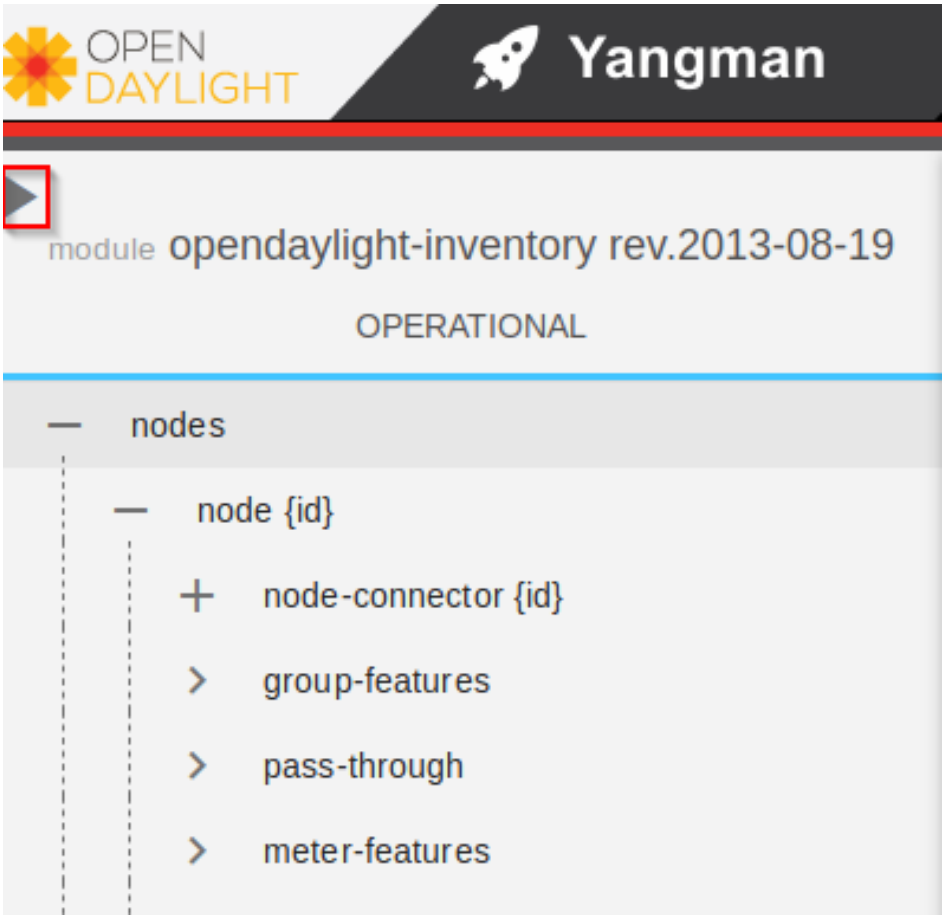

Figura 4.112: Panel Inventory Node

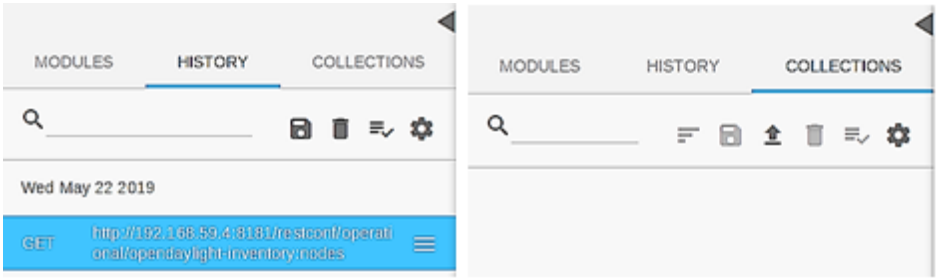

Figura 4.113: Pantallas de Historial y Colecciones respectivamente

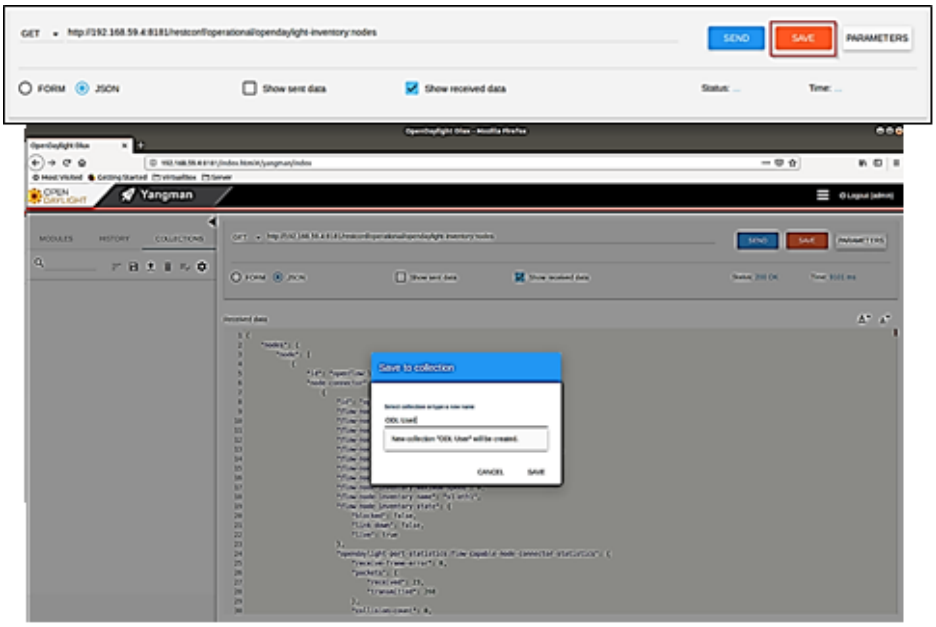

Figura 4.114: Como guardar una colección

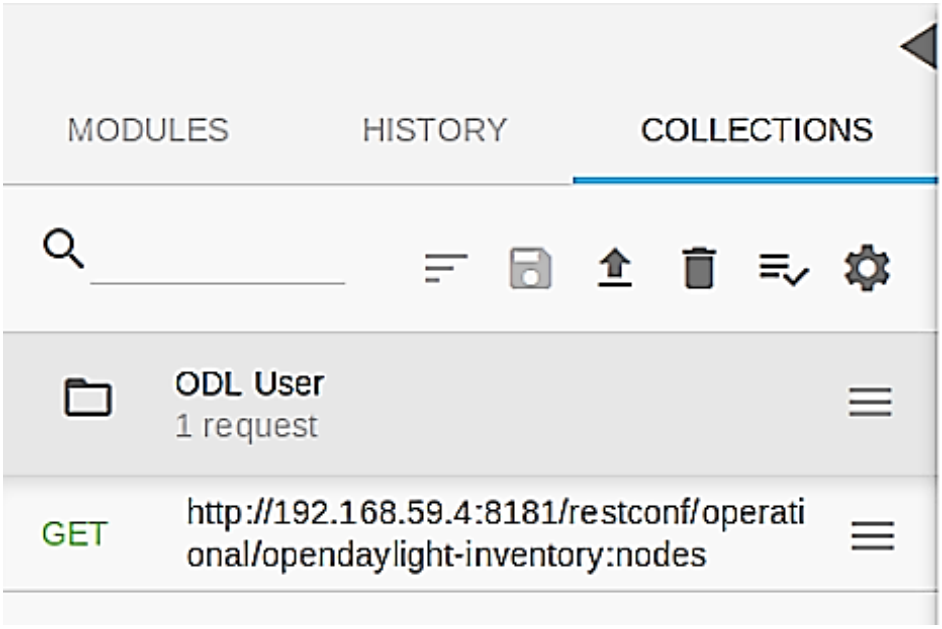

Figura 4.115: Colecciones guardadas

4.9. Simulación con OpenDaylight y OpenFlow Manager (OFM) App

En el punto anterior se vio la utilización del controlador OpenDaylight (ODL) como plataforma de desarrollo y entrega de aplicaciones de código abierto. OpenFlow Manager (OFM) [99] es una aplicación desarrollada para ejecutarse sobre ODL para visualizar topologías de OpenFlow (OF), programar rutas OF y recopilar estadísticas OF. Aquí se lo muestra como un claro ejemplo de aplicación de frontera norte, pues trabaja directamente sobre ODL (Figura 4.116).

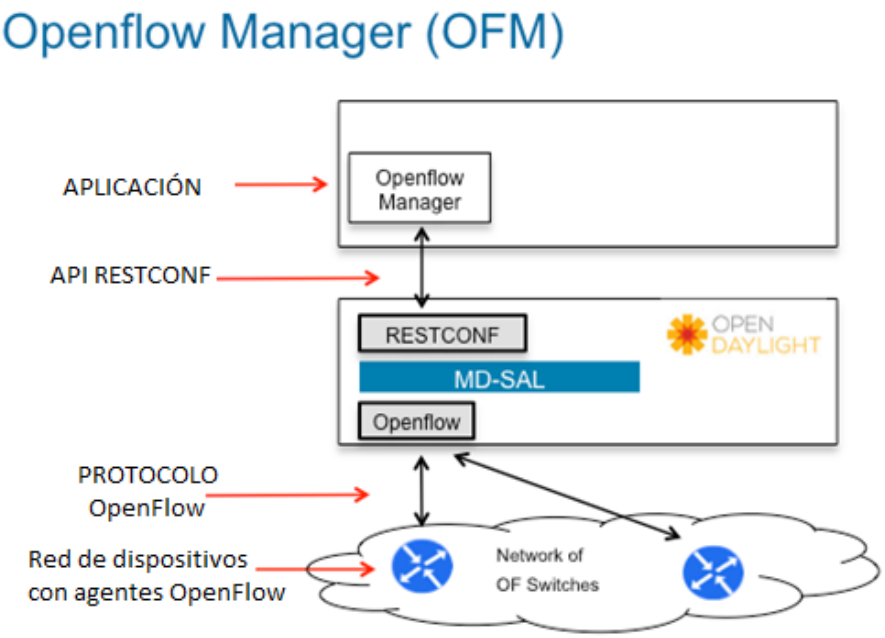

Figura 4.116: Arquitectura Openflow Manager

Dentro de ODL hay modelos YANG de la topología de red y cómo programar flujos a través de la red. La capa de adaptación de servicio dirigida por modelo (MD-SAL) toma estos modelos y genera automáticamente un conjunto de API REST (denominado RESTCONF) que las aplicaciones pueden llamar. OFM es la aplicación que llama a las API RESTCONF para recuperar el inventario del switch OpenFlow, así como para programar flujos y recopilar estadísticas. El otro componente clave aquí es la interfaz de usuario basada en HTML5 / CSS / Javascript que utiliza varios frameworks de código abierto, incluidos AngularJS y NeXt. OFM es accesible a través de un navegador web como Chrome, Firefox etc.

4.9.1. Instalación de OpenFlow Manager (OFM)

En el ejemplo que se desarrolla a continuación, se utiliza el controlador OpenDayLight versión Berillium SR4, a los efectos de facilitar la experiencia se indica brevemente su instalación en la MV ODL.

Se puede obtener Beryllium de:

https://nexus.opendaylight.org/content/repositories/public/org/opendaylight/integration/distribution-karaf/0.4.4-Beryllium-SR4/

- *$wget https://nexus.opendaylight.org/content/repositories/public/org/opendaylight/integration/distribution-karaf/0.4.4-Beryllium-SR4/distribution-karaf-0.4.4-Beryllium-SR4.tar.gz*

Luego descomprimir:

- *tar xvfz distribution-karaf-0.4.4-Beryllium-SR4*

En caso de no tener Java 8, instalar el runtime y el kit de desarrollo teclear en la terminal:

- *sudo apt install openjdk-ojdk openjdk-8-jre*

luego tipear:

- *echo $JAVA\_HOME$*
- *nano .baschrc y agregar al final la siguiente linea*
- *export $JAVA_HOME = /usr/lib/jvm/java - 8 - openjdk - amd64$*

Luego se procede a guardar, ejecutando:

- *ls /usr/lib/jvm/ se puede verificar que estén java-1.8.0-openjdk-amd64 java-8-openjdk-amd64*

Luego:

- *echo $JAVA\_HOME$*
- *export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/*

A continuación, entramos al directorio de la distribución:

- *cd distribution-karaf-0.4.4-Beryllium-SR4/*

Arrancamos el controlador con el comando

- *./bin/karaf*

Se verifican las características instaladas:

- *feature:list -i*

Se instalan las siguientes características para ODL con soporte para OpenFlow Manager (OFM):

- *feature:install odl-restconf-all odl-openflowplugin-all odl-l2switch-all odl-mdsal-all odl-yangtools-common odl-mdsal-all odl-dlux-core odl-dlux-node odl-dlux-yangui odl-dlux-yangvisualizer*

Para la instalación de OpenFlow Manager, en una nueva terminal tipeamos:

- *sudo apt-get update*
- *sudo apt-get install -y npm*
- *sudo apt-get install –y nodejs-legacy*
- *git clone http://github.com/CiscoDevNet/OpenDayLight-OpenFlow-App.git*

luego:

- *cd OpenDayLight-OpenFlow-App/*

Se edita "env.module.js", para que baseURL coincida con la dirección IP del controlador ODL (Figura 4.117)

- *sudo nano ./ofm/src/common/config/env.module.js*

Luego se instala el servidor grunt

- *sudo npm install -g grunt-cli*

Se arranca el servidor grunt (Figura 4.118) con el comando:

- *sudo grunt*

En otra terminal, utilizando NMAP localhost, se puede constatar que ODL escucha en el puerto 8181 y grunt OFM en el puerto 9000 (Figura 4.119).

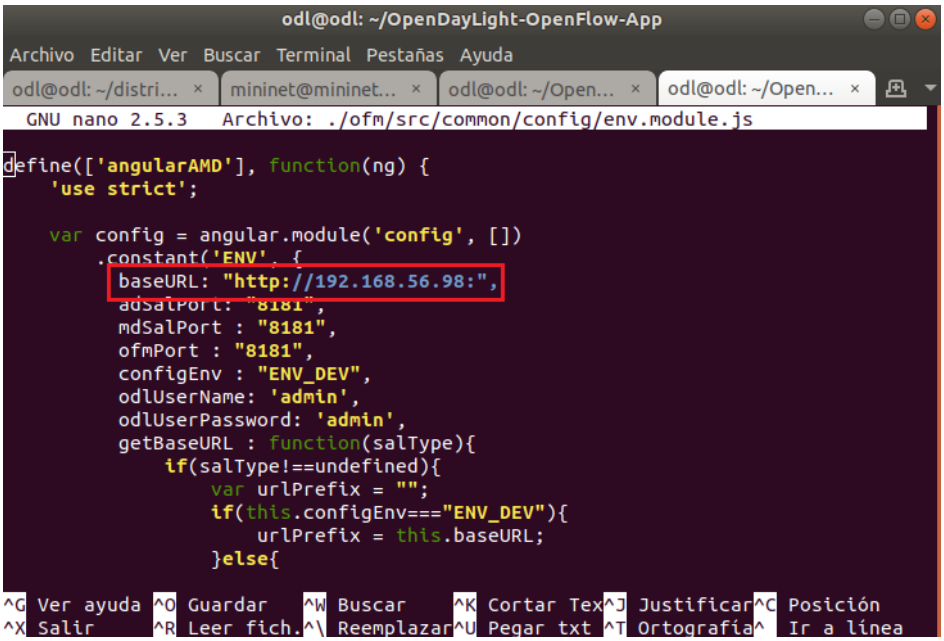

Figura 4.117: Colocando la dirección del controlador

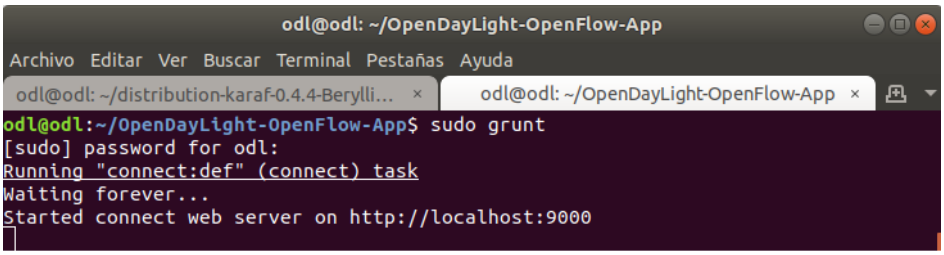

Figura 4.118: Arranque servidor grunt

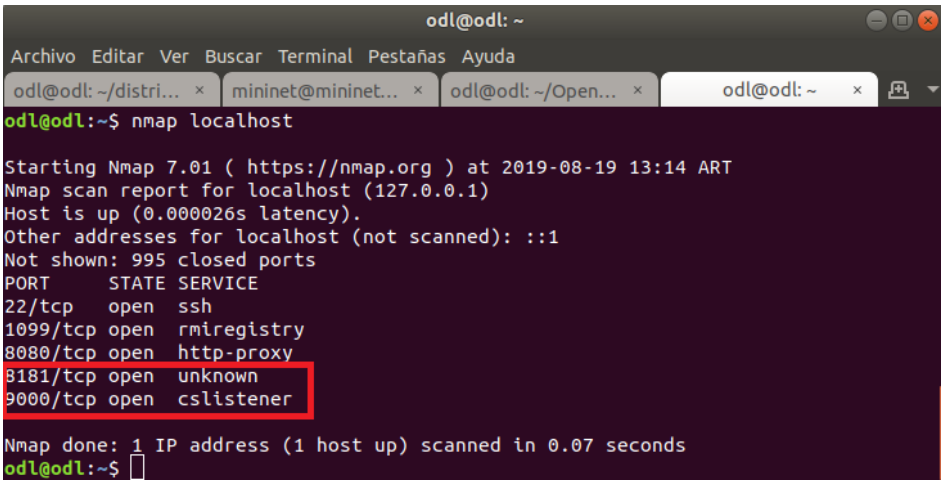

Figura 4.119: Comprobación de los puertos de ODL y grunt

Hecho esto se está en condiciones de utilizar la aplicación OFM, y a manera de ejemplo comparativo se hará con la topología de prueba utilizada en el punto 4.5.6 Administrando Flujos con Mininet.

4.9.2. Iniciando la simulación con OFM

En la MV Mininet se arranca la siguiente topología:

- *sudo mn –topo=tree,2,2 –controller=remote,ip=192.168.56.98 –mac –switch ovsk,protocols=OpenFlow13*

Con el navegador apuntando a 192.168.56.98:8181, accedemos a la interfaz Dlux en el menú Topology (Figura 4.120)

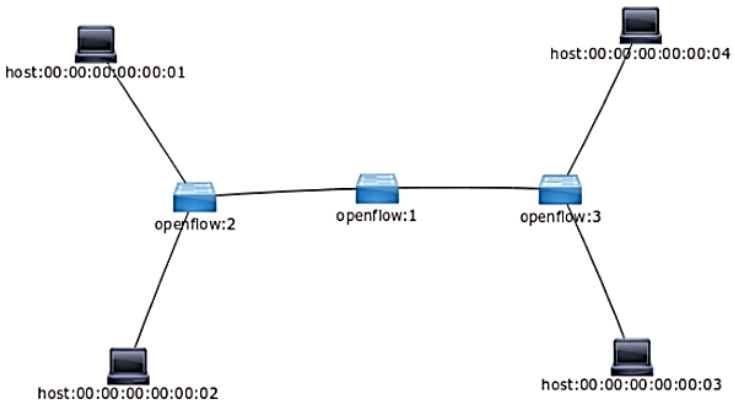

Figura 4.120: Topología de prueba en ODL Dlux

Apuntando al servidor grunt OFM, en este caso http://192.168.56.98:9000, pestaña Basic view, se muestra la topología de prueba (Figura 4.121)

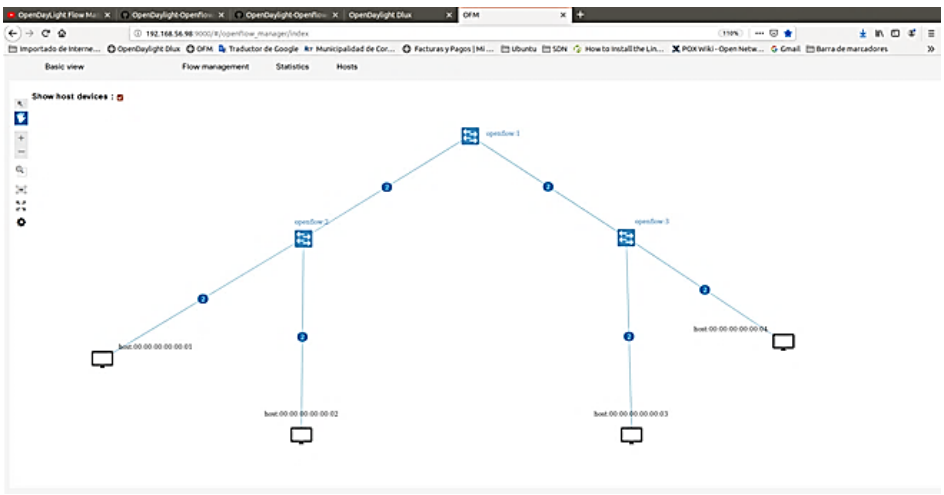

Figura 4.121: Basic view en el servidor OFM

Esta interfaz es sumamente intuitiva y no requiere de mayor explicación, como se puede apreciar en la Figura 4.121 existen cuatro pestañas: Basic view, Flow management Statistics y Hosts. En la pestaña Flow management, se procede a aplicar una regla de flujo que deniegue el tráfico del host 2, para hacer exactamente lo mismo que se desarrolló en el punto 4.5.6, para lo cual examinamos la pestaña Host verificando el puerto y dirección mac del host 2 (Figura 4.122)

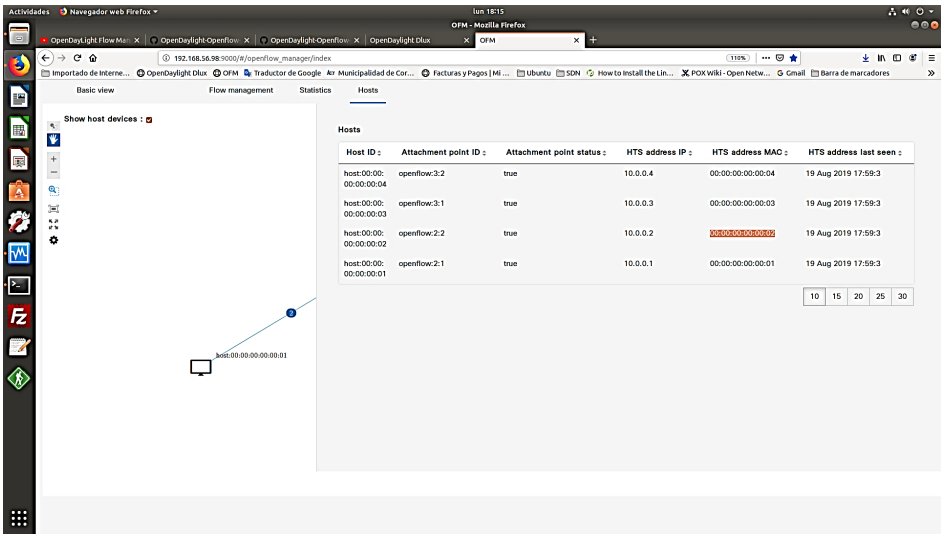

Figura 4.122: Pestaña Host mostrando los parámetros del host 2 (dirección mac resaltada)

En la terminal donde está el cli de Mininet ejecutamos el comando pingall, se observa en la salida que todos los host contestan el ping (Figura 4.125). Luego en la pestaña Flow management (Figura 4.123), se procede a agregar una regla de flujo en la tabla 0 del switch 2, aquí denominado openflow:2, y el puerto correspondiente al host 2 es openflow:2:2 (Figura 4.124)

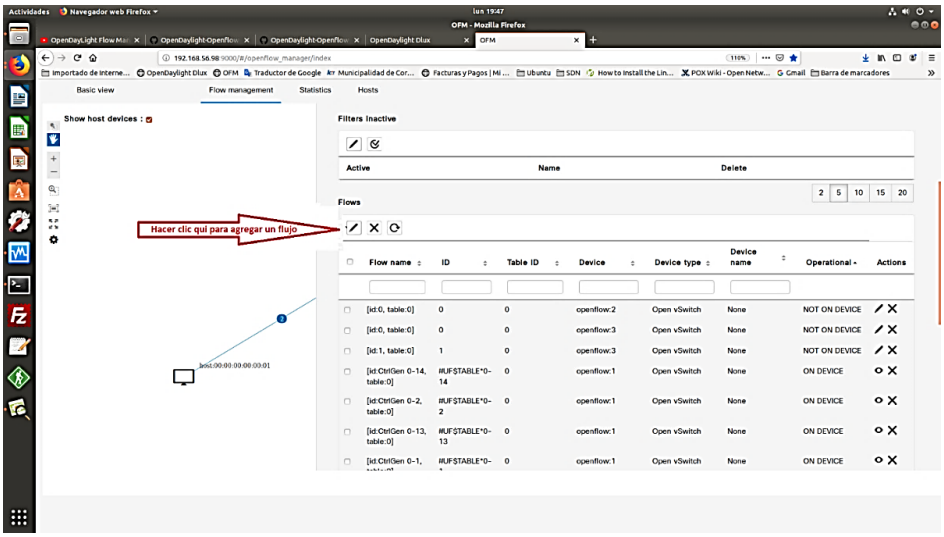

Figura 4.123: Pestaña Flow mangement

Como se observa en la Figura 4.124, la regla de flujo establece que para un "Match", correspondiente a la dirección mac del host 2, que en nuestro caso es 00:00:00:00:00:02, se establezca una "action"

"drop".

La Figura 4.125 muestra el resultado del comando pingall antes y después de aplicar la regla de flujo denegando el tráfico del host 2

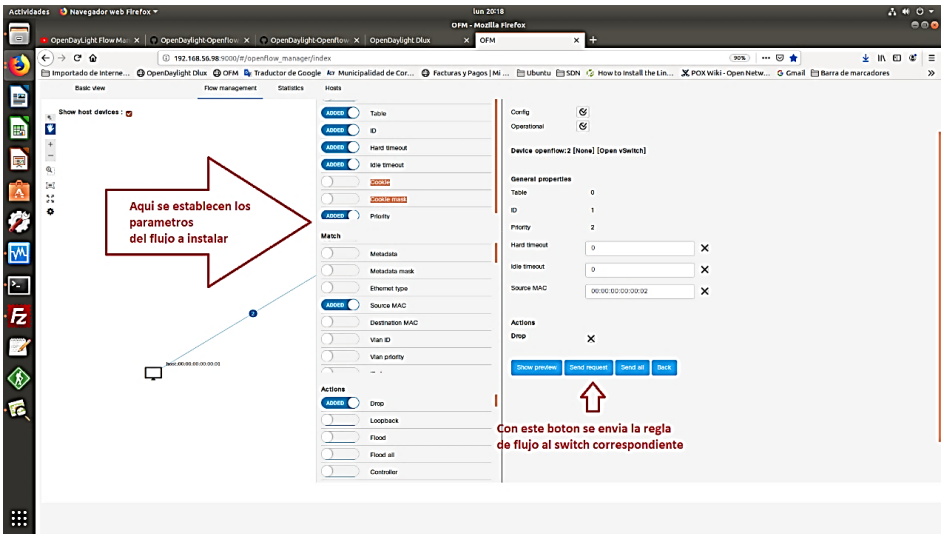

Figura 4.124: Instalando una regla de flujo con OFM

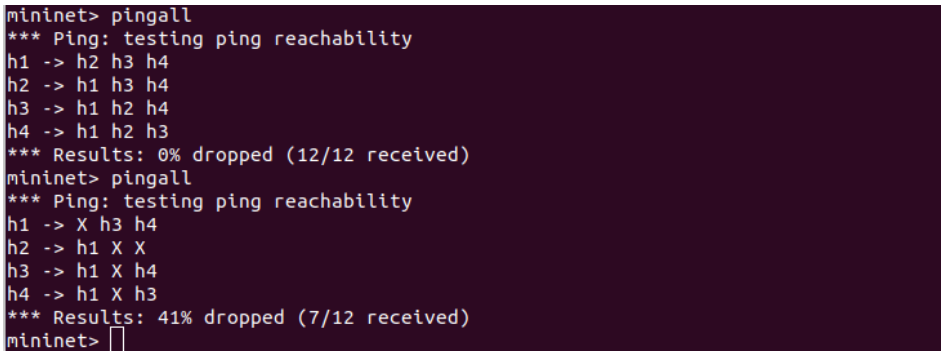

Figura 4.125: Resultado de pingall antes y después de aplicar la regla de flujo denegando tráfico a host2

En este apartado se mostró un ejemplo de aplicación frontera norte, que facilita la creación de reglas de flujo actuando directamente sobre un controlador OpenDayLigth.

4.10. Utilizar el entorno Postman con OpenDayLight

Postman [100] es un entorno de desarrollo API completo y se integra de manera flexible con el ciclo de desarrollo de software. Postman es utilizado por 7 millones de desarrolladores y más de 300,000 compañías para acceder a 130 millones de API cada mes.

Se recomienda instalar Postman en nuestro dispositivo anfitrión, pues si se instala en la MV que contiene el controlador, cada vez que cambiemos MV con otros controladores, deberíamos instalar también allí Postman, instalando en el anfitrión nos ahorramos este procedimiento. Para instalar Postman sencillamente se hace con el comando:

- *snap install postman*

Una vez instalado postman se debe crear una cuenta gratuita, o bien si ya se posee alguna cuenta de Google, se puede utilizar esta como credenciales de acceso. Una vez hecho esto se puede acceder a la interfaz de Postamn sencillamente tipeando el comando postman desde una terminal.

Con Postman, se busca que sea un poco más cómodo utilizar la API YANG UI de ODL, como ejemplo vamos a usar la misma URL de dicha API, se la puede obtener por ejemplo dirigiéndonos a YANG UI, luego opendaylight-inventory-nodes/node/ y seleccionando el nodo y tabla correspondiente podemos hacer un "Copy to clipboard", como muestra la Figura 4.126.

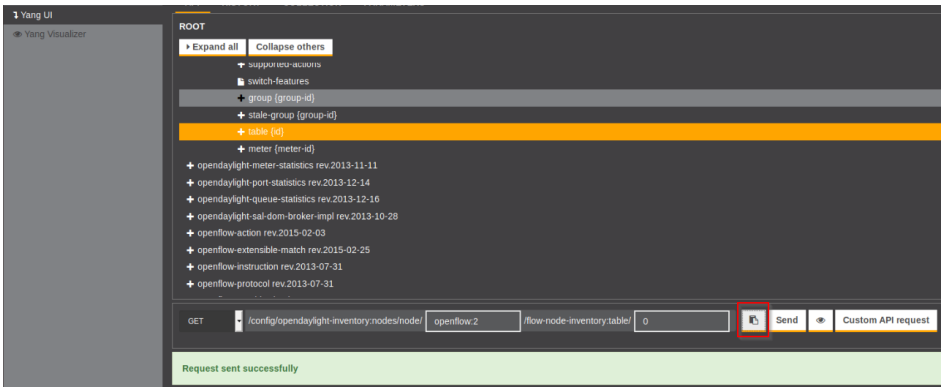

Figura 4.126: Obteniendo la URL de la API Inventory de YANG UI

Se obtiene la siguiente entrada:

http://192.168.59.10:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:2/flow-node-inventory:table/0

Para estar seguro de que existe dicha entrada, se recomienda ejecutar GET y comprobar si es exitoso.

Otra forma de realizar todo esto, es con la opción "Custom API request", donde se puede poner el código XML. Lo mismo que el punto anterior pero con Postman. Se ejecuta el comando postan desde la terminal, se hace lo que muestra la Figura 4.127

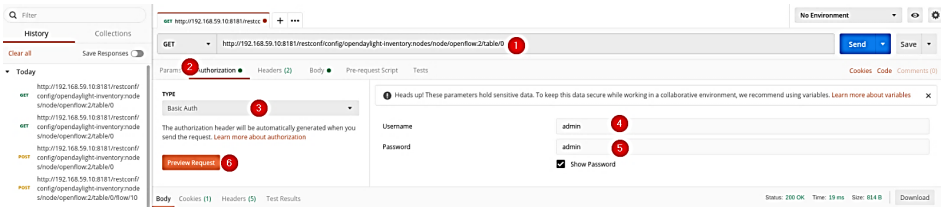

Figura 4.127: Utilizando postman

- *1. copiar la URL,*
- *2. luego ir a la pestaña Authorization,*
- *3. seleccionar el tipo Basic Auth*
- *4. colocar las credenciales de ODL DLUX (por defecto usuario y clave admin),*
- *5. por último seleccionar Preview Request para que tome la configuración.*

Luego en la pestaña Body Figura 4.128.

- *1. pestaña Body,*

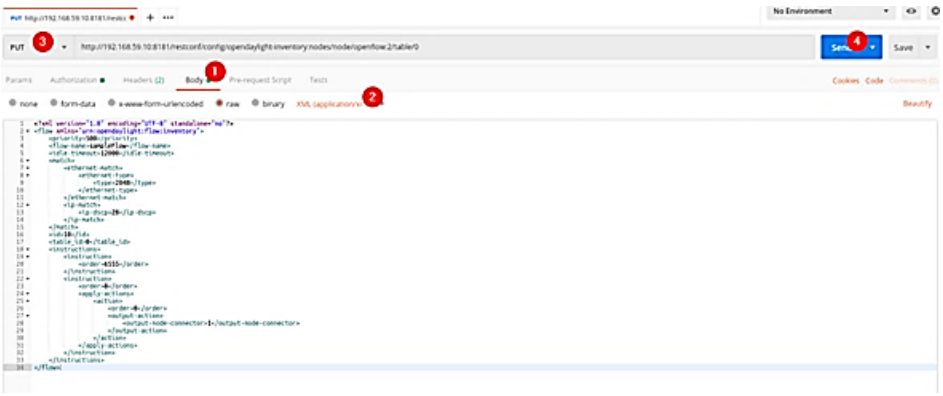

Figura 4.128: Pestaña Body de Postman

- *2. seleccionar la entrada XML (podría usarse JSON también)*
- *3. luego PUT (con la misma URL).*

Por último, antes de tocar Send se debe copiar el siguiente código xml (Figura 4.129.

Con esto se ejecuta el PUT (o POST si ya existe un flujo con el mismo nombre o id creado).

Si dejamos la misma URL y seleccionamos GET en vez de PUT, tendremos el siguiente resultado en la sección inferior del Postman (puede que sea necesario realizar un scroll hacia abajo con el mouse).

Como vemos en la Figura 4.130 con el código JSON resultante, se puede ver lo que se había incorporado en el sampleFlow definido en nuestro XML en la tabla 0.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<flow xmlns="urn:opendaylight:flow:inventory">
  <priority>500</priority>
  <flow-name>sampleFlow</flow-name>
  <idle-timeout>12000</idle-timeout>
  <match>
    <ethernet-match>
      <ethernet-type>
        <type>2048</type>
      </ethernet-type>
    </ethernet-match>
    <ip-match>
      <ip-dscp>28</ip-dscp>
    </ip-match>
  </match>
  <id>10</id>
  <table_id>0</table_id>
  <instructions>
    <instruction>
      <order>6555</order>
    </instruction>
    <instruction>
      <order>0</order>
      <apply-actions>
        <action>
          <order>0</order>
          <output-action>
            <output-node-connector>1</output-node-connector>
          </output-action>
        </action>
      </apply-actions>
    </instruction>
  </instructions>
</flow>
```

Figura 4.129

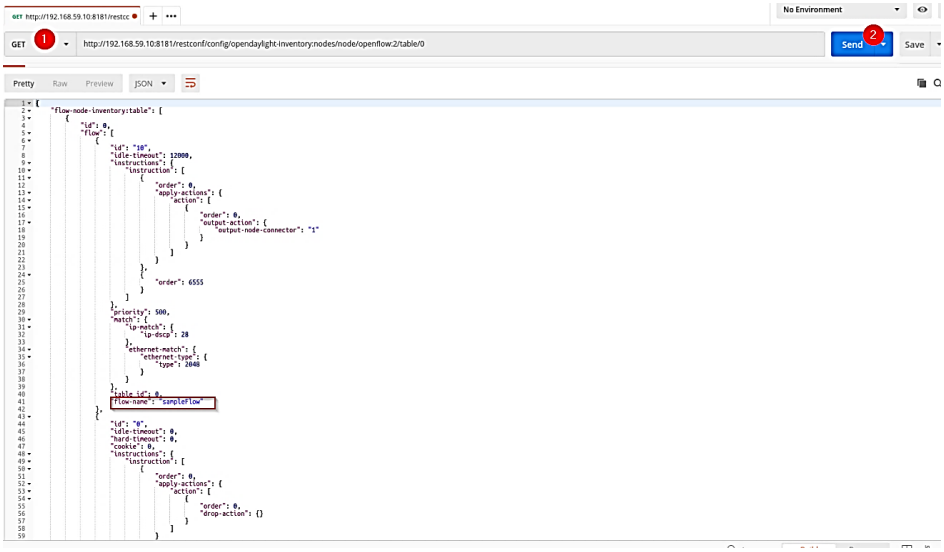

Figura 4.130: Código JSON del flujo resultante

Capítulo 5

SDN en un contexto de Ingeniería de Tráfico y QoS

5.1. El contexto

La Unión Internacional de Telecomunicaciones (UIT), menciona en su Recomendación E.600, "La ingeniera de tráfico tiene por objeto lograr los objetivos de calidad de la aptitud para cursar tráfico para servicios de telecomunicaciones. La calidad de la aptitud para cursar tráfico es uno de los principales factores de la calidad de servicio".

La Recomendación UIT-T E.800 facilita una serie de los términos más comúnmente utilizados en el estudio y la gestión de la calidad de servicio (QoS). La definición de los términos tiene por contexto la QoS. En otros contextos, algunos de los términos podrían definirse de manera diferente. Por consiguiente, habrá de tomarse la precaución de utilizar los términos en el contexto adecuado. Dicho esto, se define la Calidad de Servicio (QoS) como:

"La totalidad de las características de un servicio de telecomunicaciones que determinan su capacidad para satisfacer las necesidades explícitas e implícitas del usuario del servicio".

Mediante el uso de mecanismos de calidad de servicio (QoS), los administradores de red pueden usar los recursos existentes de manera eciente y garantizar el nivel de servicio requerido por usuarios y aplicaciones, sin sobredimensionar sus redes. El concepto de calidad de servicio es aquel en el que los requisitos de algunas aplicaciones y usuarios son más exigentes en cuanto a recursos de la red, que otros. Esto signica que ése tráco necesita un tratamiento preferencial, sin que eso signifique imposibilitar el tráfico de usuarios y aplicaciones menos exigentes. Se puede definir QoS como un conjunto de tecnologías para diferenciar la asignación de recursos en una red entre distintas clases de servicio, proveyendo control de congestión, bajo retardo y máxima utilización de la capacidad disponible. Srinivasnan Keshav", en su libro "An Engineering Approach to Computer Networking" [101], tiene una frase que resulta particularmente descriptiva de lo que pretende la ingeniería de tráfico y los mecanismos de QoS:

"El Santo Grial de las redes de computadores es diseñar una red que tenga la flexibilidad y el bajo costo de la Internet, pero que ofrezca las garantías de calidad de servicio

extremo a extremo de la red telefónica".

Desde la perspectiva de la Ingeniería de Tráfico, el ruteo IP no es suficiente para atender las necesidades de calidad de servicio (QoS) requeridas por usuarios y aplicaciones de las redes modernas. La principal característica de ruteo IP es que se basa en la dirección de destino y las decisiones de ruta se toman en cada nodo conforme una tabla de conmutación que se construye a partir de métricas elementales o combinación de métricas (retardo, ancho de banda, carga, confiabilidad, MTU), en un valor denominado "costo". Las rutas se eligen y crean en la tabla de ruteo basada en la distancia administrativa del protocolo de ruteo. Las rutas aprendidas del protocolo de ruteo con la mínima distancia administrativa están instaladas en la tabla de ruteo. Si hay trayectos múltiples hacia el mismo destino desde un solo protocolo de ruteo, entonces, los trayectos múltiples tendrían la misma distancia administrativa y se elige el mejor trayecto basándose en las métricas. La métrica (costo) son valores asociados con rutas específicas, clasificándolos de los más a los menos preferidos. Los parámetros usados para determinar la métrica diferencian para diversos protocolos de ruteo. Se selecciona el trayecto de menor costo como trayecto óptimo y se instala en la tabla de ruteo por mencionar un ejemplo muy ubicuo; el protocolo de ruteo OSPF utiliza generalmente como métrica el ancho de banda y aplica el algoritmo SPF (Shorted Path First) de camino más corto para construir la tabla de ruteo. Como resultado de esto todos los paquetes que compartan el mismo prefijo de red de destino seguirán el mismo camino. El enrutamiento de la ruta más corta no siempre utiliza las diversas conexiones disponibles en la red. De hecho, el tráfico a menudo se distribuye de manera desigual a través de la red, lo que puede crear puntos de congestión (cuellos de botella), mientras que otras partes de la red pueden estar con muy poca carga o subutilizadas. Otro ejemplo en cuanto a esto es el protocolo de ruteo BGP, que es un ruteo basado principalmente en políticas y eso torna más complejo el tema de aplicación de Ingeniería de Tráfico y QoS en esos entornos de red.

La comunidad de Internet ha realizado inmensos esfuerzos para abordar el problema y ha desarrollado una serie de tecnologías para mejorar Internet con capacidades de QoS. Podemos citar tecnologías que han surgido en los últimos años como los componentes básicos para habilitar QoS en Internet. Las arquitecturas y los mecanismos desarrollados en estas tecnologías abordan dos problemas clave de QoS en Internet:

- *asignación de recursos.*
- *optimización del rendimiento.*

Se han desarrollado y estandarizado dos mecanismos de QoS:

- *1. IntServ (Integrated Services - ISA) y protocolo RSVP. Fuertemente influido por ATM y las tecnologías telefónicas. El usuario solicita de antemano los recursos que necesita; cada router del trayecto ha de tomar nota y efectuar la reserva solicitada.*
- *2. DiffServ (Differentiated Services). El usuario marca los paquetes con un determinado nivel de prioridad; los routers van agregando las demandas de los usuarios y propagándolas por el trayecto. Esto le da al usuario una confianza razonable de conseguir la QoS solicitada.*

Los servicios integrados (IntServ) y los servicios diferenciados (DiffServ), son dos arquitecturas de asignación de recursos para Internet. Los nuevos modelos de servicio propuestos en ellos hacen posible la garantía de los recursos y la diferenciación del servicio para los flujos de tráfico y los usuarios.

La conmutación de etiquetas multiprotocolo (MPLS) mencionada en el capítulo 1, y la ingeniería de tráfico, por otro lado, brindan a los proveedores de servicios un conjunto de herramientas de

administración para el aprovisionamiento de ancho de banda y la optimización del rendimiento; sin ellos, sería difícil soportar QoS a gran escala y a un costo razonable. Las redes SDN aportan una visión superadora de esta problemática, reutilizando estas tecnologías con las facilidades administrativas que proporciona la programabilidad de la red y la administración de esta mediante el protocolo OpenFlow.

5.1.1. El problema del pez

Un problema típico en ingeniería de tráfico es el conocido como "problema del pez" Figura 5.1. Supongamos que un ISP trata de ofrecer servicios de diferente calidad a sus usuarios, por ejemplo, servicio premium con alta calidad y servicio normal con calidad "best effort". Supongamos además que dos usuarios (A y B) contratan los servicios premium y normal, respectivamente, en un mismo POP (Point Of Presence), y que ambos están interesados en enviar tráfico al usuario C, conectado a otro POP de la red del ISP. El router del ISP al encaminar los paquetes de A y de B hacia C no podrá en principio discriminar cuales pertenecen a A y cuales, a B, ya que el encaminamiento se realiza en base a la dirección de destino, no a la dirección de origen. Por tanto, el tráfico de A como el de B se encaminarán por la misma ruta, recibiendo probablemente ambos clientes la misma calidad de servicio aun cuando pagan tarifas diferentes. Una posible solución a este problema es lo que se denomina "policy routing", es decir encaminamiento basado en criterios que no se limitan a la dirección de destino, sino que toman en cuenta otros factores, como la dirección de origen. Sin embargo, el policy routing tienen un problema serio: los fabricantes diseñan sus equipos para la conmutación rápida, generalmente por hardware, de paquetes IP basándose exclusivamente en la dirección de destino, y siempre que se aplica policiy routing el rendimiento decae de manera alarmante.

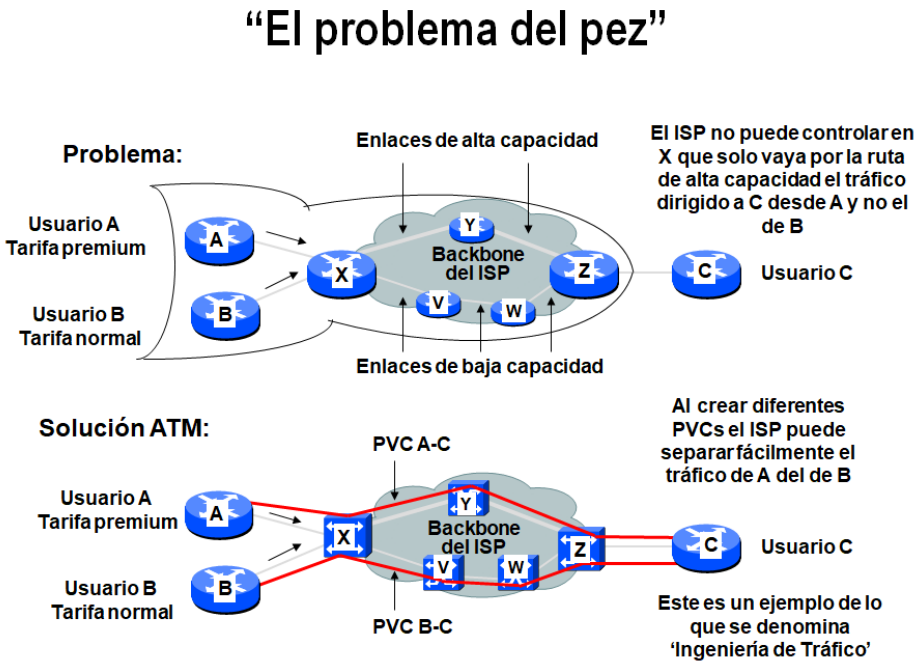

Figura 5.1: El clásico "problema del pez" en QoS

5.1.1.1. Solución ATM

Una mejor solución sería que el ISP sustituyera su red por un backbone ATM. En este caso sería fácil constituir dos PVC independientes para encaminar el tráfico de A por la ruta de alta capacidad (la superior) y el de B por la de baja capacidad (Figura 5.1 parte inferior).

5.1.1.2. Solución MPLS

Una solución que mejora el escenario anterior es una red MPLS (Figura 5.2), donde la conmutación se realiza en base a etiquetas y el proveedor tiene la libertad de marcar los paquetes que ingresan a su red, clasificarlos y establecer la mejor ruta en función de la asignación de recursos representada por la etiqueta.

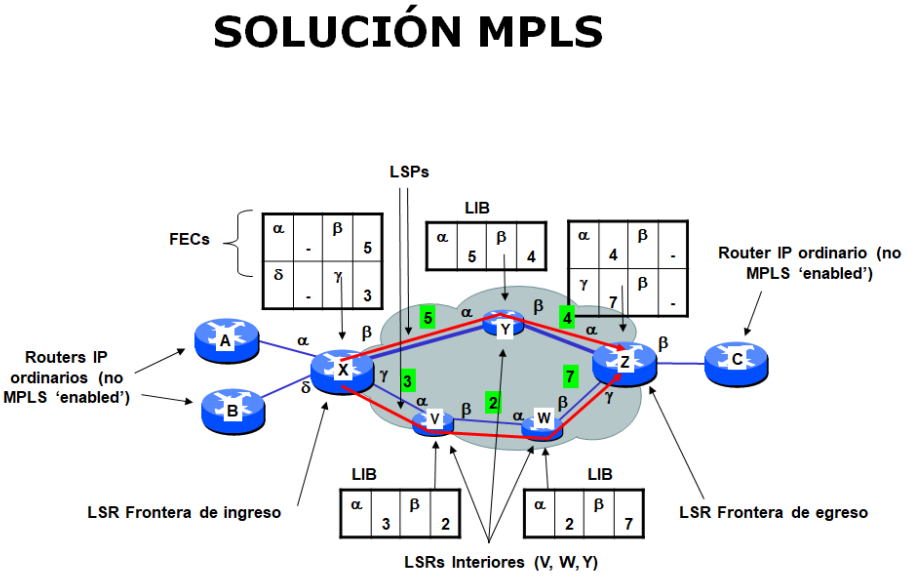

Figura 5.2: Solución MPLS al "problema del pez"

Terminología de la Figura 5.2

- *FEC (Forwarding Equivalence Class): conjunto de paquetes que entran en la red MPLS por la misma interfaz, que reciben la misma etiqueta y por tanto circulan por un mismo trayecto. Normalmente se trata de datagramas que pertenecen a un mismo flujo. Una FEC puede agrupar varios flujos, pero un mismo flujo no puede pertenecer a más de una FEC al mismo tiempo.*
- *LSP (Label Switched Path): camino que siguen por la red MPLS los paquetes que pertenecen a la misma FEC. Es equivalente a un circuito virtual en ATM o Frame Relay.*
- *LSR (Label Switching Router): router que puede encaminar paquetes en función del valor de la etiqueta MPLS*
- *LIB (Label Information Base): La tabla de etiquetas que manejan los LSR. Relaciona la pareja (interfaz de entrada - etiqueta de entrada) con (interfaz de salida - etiqueta de salida) Los LSR pueden ser a su vez de varios tipos:*

- *LSR Interior: el que encamina paquetes dentro de la red MPLS. Su misión es únicamente cambiar las etiquetas para cada FEC según le indica su LIB*
- *LSR Frontera de ingreso: los que se encuentran en la entrada del flujo a la red MPLS (al principio del LSP). Se encargan de clasificar los paquetes en FECs y poner las etiquetas correspondientes.*
- *LSR Frontera de egreso: Los que se encuentran a la salida del flujo de la red MPLS (al final del LSP). Se encargan de eliminar del paquete la etiqueta MPLS, dejándolo tal como estaba al principio.*

Creación de los LSP (Label Switched Path)

- *El establecimiento de la ruta se puede hacer:*
- *Por configuración, de forma estática (equivalente a los PVCs en ATM)*
- *Por un protocolo de señalización:*
  - *LDP: Label Distribution Protocol*
  - *RSVP mejorado*
- *El enrutamiento del LSP se hace en base a la información que suministra el protocolo de routing, p.ej. OSPF o IS-IS*
- *Siempre se usan algoritmos del estado del enlace, que permiten conocer la ruta completa*

Sin embargo, una vez fijado el LSP falla algún enlace hay que crear un nuevo LSP por otra ruta para poder pasar tráfico, por ende, se está en un escenario de administración que requiere atención en el nodo específico. Aquí es donde las redes SDN presentan un contexto superador, estableciendo el manejo de las etiquetas desde el controlador.

5.2. MPLS Ingeniería de Tráfico y SDN

Utilizar tecnología SDN en un entorno MPLS puede reducir la complejidad en el plano de control dinámico y permitir ingeniería de tráfico con QoS para la creación de servicios flexibles y dar servicio a usuarios con distintos requerimientos de QoS.

Los protocolos como OSPF-TE [102], RSVP-TE [103], LDP [104], LMP [105], I-BGP [106] y MP-BGP [107] requieren operaciones del plano de control distribuido y dinámico, están obligados a encontrar rutas disponibles, reservar ancho de banda para crear LSP (Label Switched Path) con ingeniería de tráfico, monitorear las rutas con el fin de mantener las garantías, distribuir etiquetas a los ruteadores a lo largo de un camino, y establecer rutas entre sistemas autónomos. Los ruteadores de la red deben tener suficiente capacidad de memoria y de CPU para soportar todos estos protocolos y sus correspondientes procesos. Está claro que cuando alguno de ellos cambia o se actualiza se debe configurar el dispositivo.

Suponiendo que los dispositivos soporten OpenFlow, un cambio a SDN eliminaría la necesidad de implementaciones de protocolos residentes en los ruteadores. Un controlador OpenFlow, por ejemplo, podría contener las reglas de flujo para el establecimiento de rutas en la red. Las aplicaciones Northbound implementarían las funciones que actualmente presta el código de protocolo residente en el ruteador, y en ese caso, los ruteadores ya no requieren de tanta capacidad, reduciendo así las necesidades de recursos y también de energía. Se puede implementar servicios escribiendo código para el software

de aplicación de la interfaz con el controlador OpenFlow. Hacer todo esto sin SDN, o añadir una característica a un protocolo estandarizado requeriría emitir una propuesta a la IETF, seguido de un período de discusión y refinamiento antes de que una nueva versión del protocolo sea publicada; por ende, requeriría de un gran esfuerzo además de llevar mucho tiempo para su implementación.

5.3. Ejemplos de aplicación

A continuación, se muestra algunos ejemplos de aplicación de los recursos SDN mostrados en los capítulos anteriores. Este capítulo no pretende ser un compendio de las tecnologías aplicadas en Ingeniería de Tráfico y QoS pues el lector comprenderá que la complejidad y extensión de dichas tecnologías son una disciplina en si mismas, sencillamente se intenta mostrar cómo se pueden implementar de manera rápida y eficiente, funcionalidades que resultan de compleja implementación con las tecnologías tradicionales

5.3.1. Una solución al problema del pez con SDN

5.3.1.1. Descripción del escenario de simulación

Se utiliza Mininet para implementar una topología tipo "pez" Figura 5.3 perteneciente a un hipotético ISP y efectuar la simulación implementando dos rutas, una para clientes "premiun" con alta demanda de recursos y otra para clientes "estándar", con menores requerimientos.

El dispositivo s1 concentra todos los clientes premiun mientras que el dispositivo s2 concentra todos los clientes estándar, ambos dispositivos se conectan al s3 del ISP a partir del cual inician su tránsito en la red del proveedor que cuenta con las siguientes características:

- *Ruta s3 – s6 – s7, cuenta con enlaces de un Gbps y latencia de 5 ms.*
- *Ruta s3 – s4 – s5 – s7 tiene enlaces de 100 Mbps y latencia de 20 ms.*

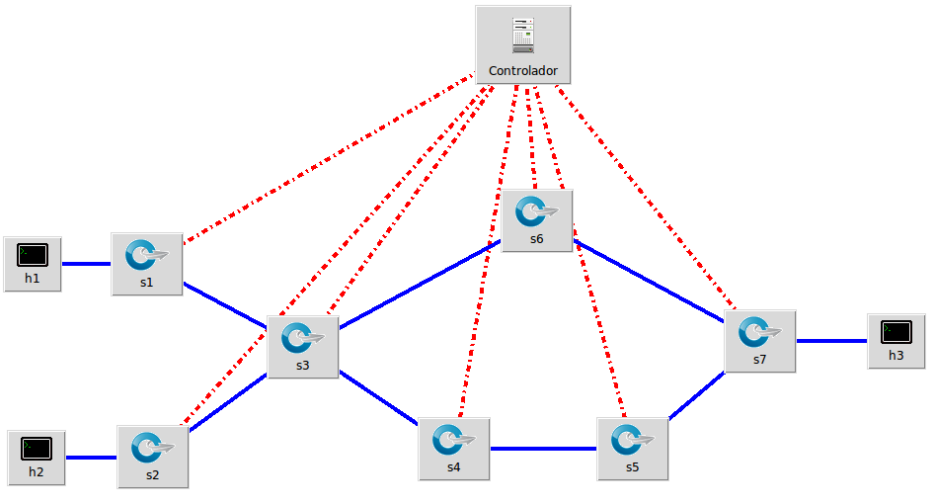

Figura 5.3: Topología tipo "pez" para simulación de rutas

La Figura 5.4 muestra cómo se implementan las características de los enlaces en la script de Python de la topología de la Figura 5.3.

Figura 5.4: Características de los enlaces de la topología de simulación

Definida la topología, y las características de los enlaces, es evidente que, ante un escenario de ruteo convencional, la totalidad del tráfico utilizaría la ruta s3 – s6 – s7 pues es la que tiene mejores "métricas".

La idea de la simulación es cursar tráfico sin aplicar reglas de flujo y luego aplicando reglas de flujo desde el controlador SDN, y en ambos casos hacer una prueba de performance utilizando la herramienta "iperf", la cual se encuentra presente en la MV Mininet, y luego comparar los resultados.

5.3.1.2. Puesta en funcionamiento de la red SDN

Utilizando la misma metodología del punto 4.5.1 visto en el capítulo 4; se procede a arrancar las máquinas virtuales que tienen el controlador ODL, el entorno Mininet respectivamente.

En la MV que tiene el controlador ODL, qes este caso se trata de la distribution-karaf-0.4.4-Beryllium-SR4, se hace un cambio de directorio al que contiene la distribución y se arranca la aplicación karaf con el siguiente comando:

- *./bin/karaf (Figura 5.5)*

Esto habilita tener disponible la interfaz Dlux y las distinta aplicaciones en el navegador apuntando a la dirección IP de la MV ODL, puerto 8181.

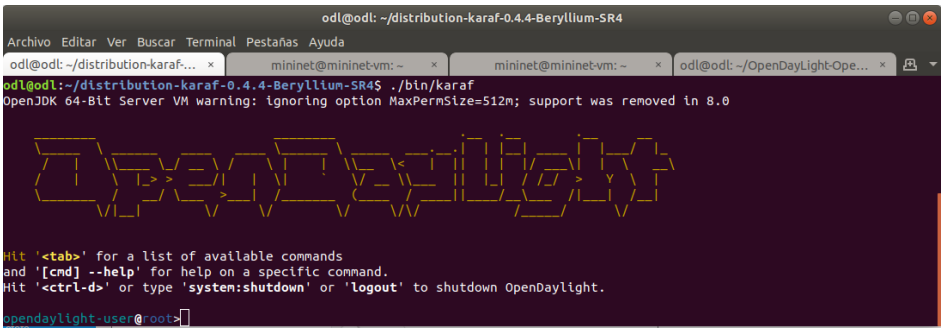

Figura 5.5: Arranque karaf

Luego en la misma MV nos situamos en el directorio que contiene el OpenFlow Manager y se arranca el servidor grunt, tal como muestra la Figura 5.6.

Hecho esto se procede la lanzar la topología de simulación en la MV Mininet (Figura 5.7), ejecutando la script de la topología de la Figura 5.3.

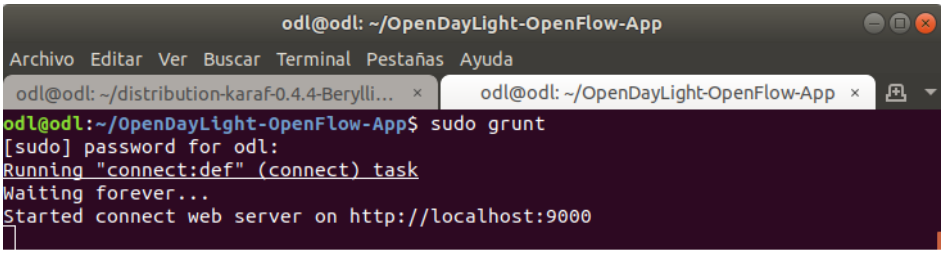

Figura 5.6: Arranque de grunt

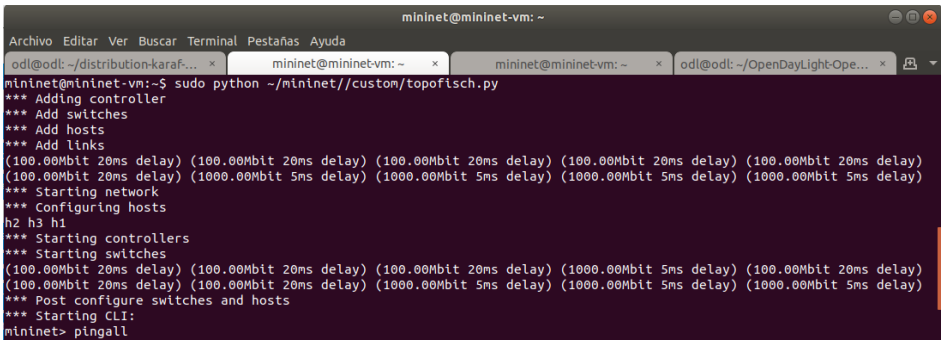

Figura 5.7: Arranque topología tipo “pez “para la simulación

Se puede verificar el funcionamiento de la red y su conectividad ejecutando el comando pingall (Figura 5.7 parte inferior).

Se procede a la identificación de interfaces y nodos con el comando net (Figura 5.8)

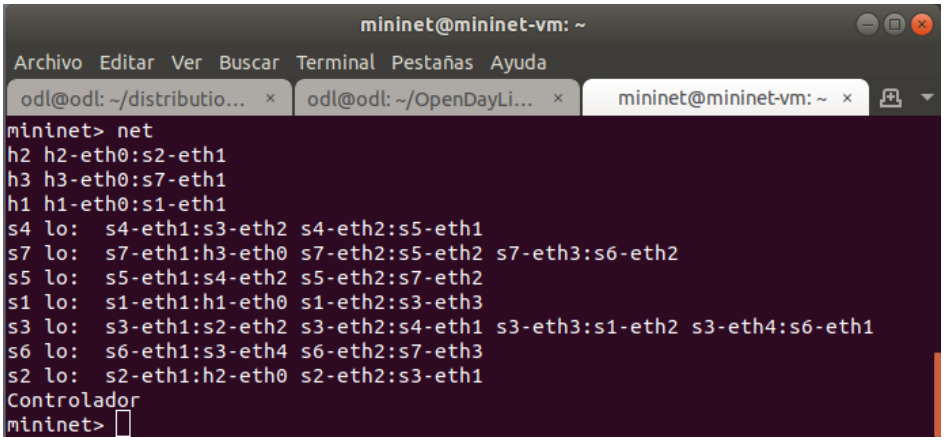

Figura 5.8: Comprobación de nodos e interfaces de la topología

Esto resulta de mucha utilidad pues ayuda en la identificación de interfaces cuando se está trabajando con OpenDayLight y OFM en distintas ventanas del navegador, tal como se muestra se muestra a continuación.

En el navegador, con la ip de la MV ODL y puerto 8181, se puede visualizar la topología en la pestaña Topology en Dlux (Figura 5.9)

Como muestra la Figura 5.9, se pueden identificar los nodos y las conexiones de la red SDN simulada. Para una mejor identificación de los puertos se utiliza el menú nodes (Figura 5.10).

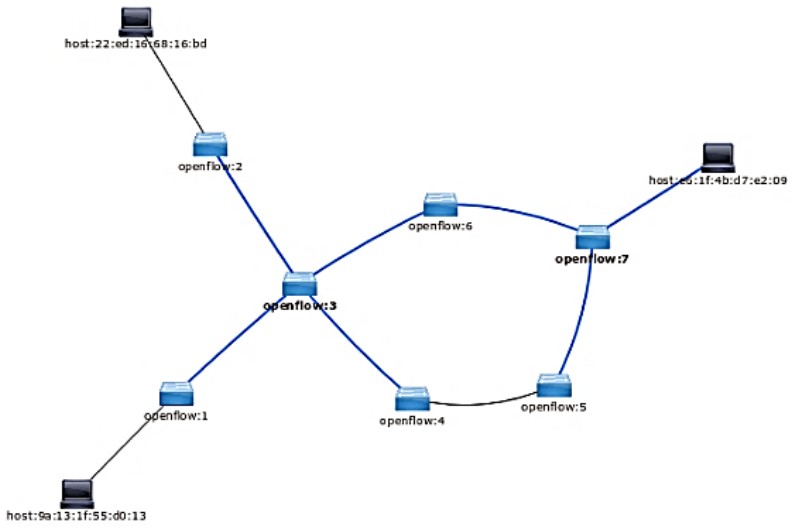

Figura 5.9: Topología de simulación en la interfaz gráfica Dlux; pestaña Topology

Esta información junto con la que brinda el comando net del cli de Mininet, es útil para identificar sin equívocos los nombres de las interfaces a la que están conectados todos los dispositivos de la red.

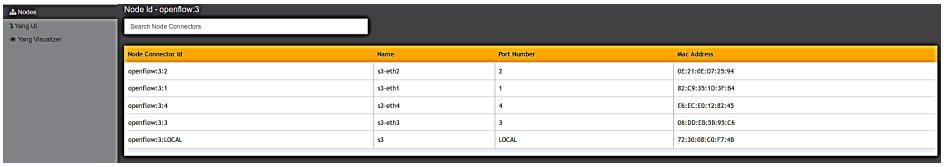

Figura 5.10: Menu Nodes de ODL

También es posible visualizar la topología en la pestaña Basic View del OpenFlow Manager (OFM), utilizando el navegador con la ip de la MV ODL, puerto 9000 (Figura 5.11)

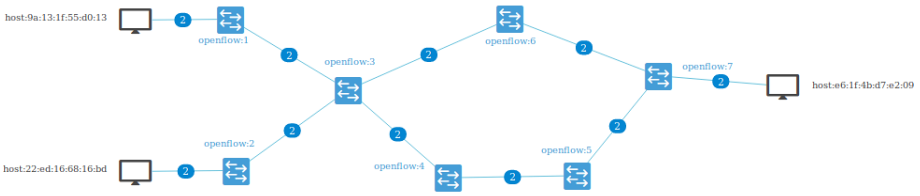

Figura 5.11: Topología “pez”, utilizada en la simulación vista en la pestaña Basic View del OFM

Para comenzar el análisis se verifican los flujos instalados en el dispositivo openflow:3 tal como muestra la Figura 5.12.

Como se puede apreciar en la Figura 5.12 cada flujo tiene como salida posible dos puertos, por tal motivo, los paquetes seguirán la ruta que les indique la tabla de ruteo que contenga el dispositivo, y que como se comentó en la descripción del escenario en el punto 5.3.1.1, siempre sería la “mejor” ruta , instalada en dicha tabla por el protocolo correspondiente.

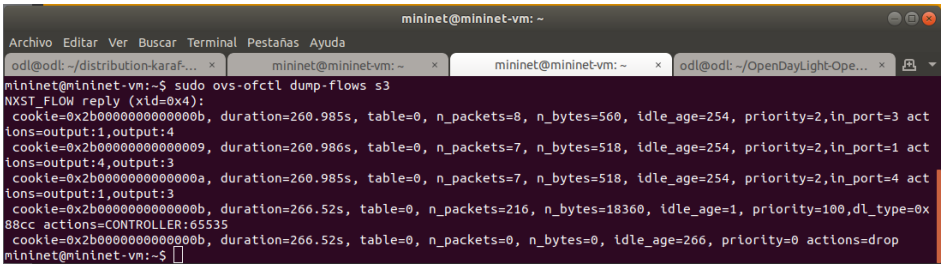

Figura 5.12: Flujos instalados inicialmente en el dispositivo openflow:3

En estas condiciones inicialmente se hace una prueba elemental de performance utilizando la aplicación iperf [108], se dice elemental pues iperf3, que es la última versión de esta excelente aplicación permite ajustar muchísimos parámetros de QoS, sin embargo, el concepto es exactamente el mismo, tanto para un escenario complejo como uno simple. Para esta simulación solo se pretende mostrar el potencial de SDN para cambiar dinámicamente el comportamiento de una red con pocas operaciones de administración.

5.3.1.3. Pruebas de performance

Para efectuar las pruebas de performance con iperf, en el cli de Mininet se abren cuatro terminales, una en cada host de origen h1 y h2 y dos en h3 (Figura 5.13), a los efectos de correr instancias de iperf en cada una de ellas.

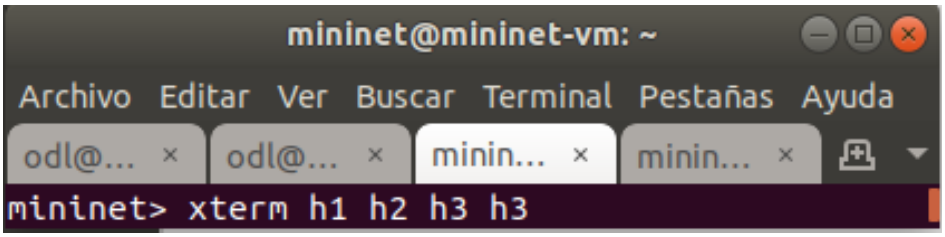

Figura 5.13: Abriendo terminales gráficas en cada host

Una vez abiertas las terminales, (tendremos una en cada ventana), se procede de la siguiente forma:

En la terminal del host 1, se corre una instancia cliente de iperf apuntando a una instancia servidor que está corriendo en el host 3 puerto 5566 (Figura 5.14), el cliente apunta a la ip del host 3, en este caso es 10.0.0.5 que está escuchando peticiones en el puerto 5566 y la opción -t 15 indica que mostrará 15 salidas una por segundo tal como está configura do el servidor en host 3 (Figura 5.16)

En la terminal del host 2, se corre una instancia cliente de iperf apuntando a una instancia servidor que está corriendo en el host 3 puerto 6677 (Figura 5.15)

En las terminales del host 3 se corren dos instancias servidor de iperf una en el puerto 5566, que atenderá peticiones del host1, y otra en el puerto 6677 que atenderá peticiones del host 2 como muestra la Figura 5.16, el servidor está configurado para escuchar peticiones en los puertos 5566 y 6677 respectivamente, y la opción -i 1 indica que imprime una salida por segundo

En la Figura 5.17 podemos ver el troughput de h1 – h3 (recuadro rojo) y la Figura 5.18 se puede ver el troughput de h2 – h3 (recuadro rojo)

Con estos resultados, resulta evidente, que el tráfico estándar no recorre la ruta prevista para

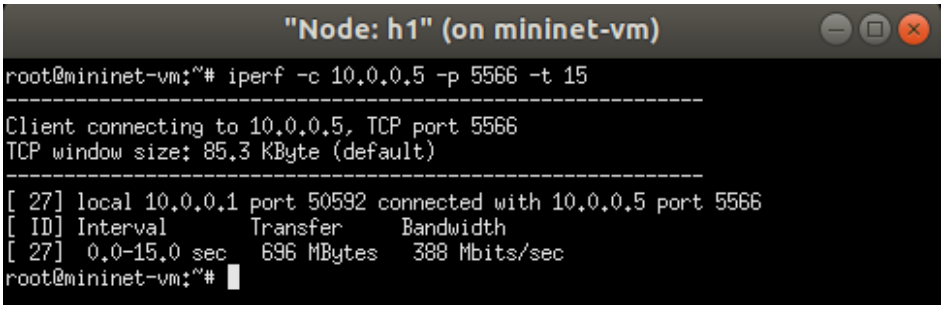

Figura 5.14: Instancia cliente iperf en nodo h1

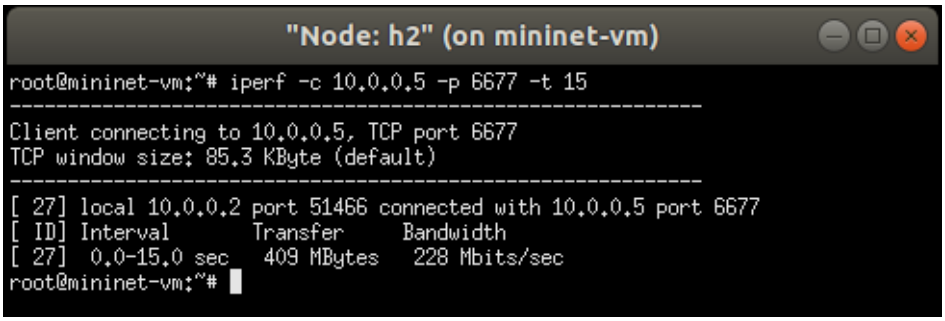

Figura 5.15: Instancia cliente iperf en nodo h2

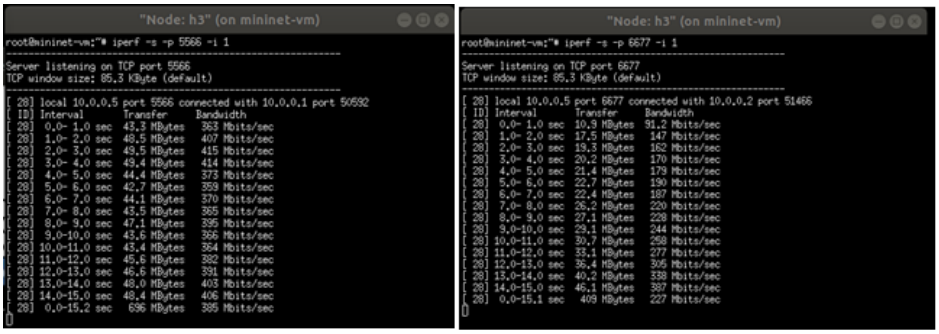

Figura 5.16: Instancias servidor iperf en h3

Figura 5.17: Troughput h1 – h3 con flujos iniciales

Figura 5.18: Troughput h2 – h3 con flujos iniciales

el mismo, pues se observa un troughput muy superior al previsto para dicho tráfico, y esto también interfiere con el troughput previsto para el tráfico premium.

En estas condiciones, se procede a definir nuevas reglas de flujo en el dispositivo openflow:3, de manera que todo el tráfico proveniente del dispositivo openflow:1 se conmute por la interfaz que conecta el dispositivo openflow:3 con el dispositivo openflow:6. Y todo el tráfico que proviene del dispositivo openflow:2, se conmute por la interfaz que conecta el dispositivo openflow:3 con el dispositivo openflow:4. Con criterio similar se definen los caminos para los dispositivos openflow:6; openflow:4, openflow:5 respectivamente, creando de esta forma dos caminos independientes, uno para los clientes premium y otro para los clientes estándar.

La Figura 5.19 muestra los flujos instalados en las condiciones iniciales en las que se hizo la prueba mostrada en el punto anterior y el recuadro rojo muestra los flujos que deben ser redefinidos para que el dispositivo openflow:3 atienda debidamente las demandas del tráfico premium y estándar.

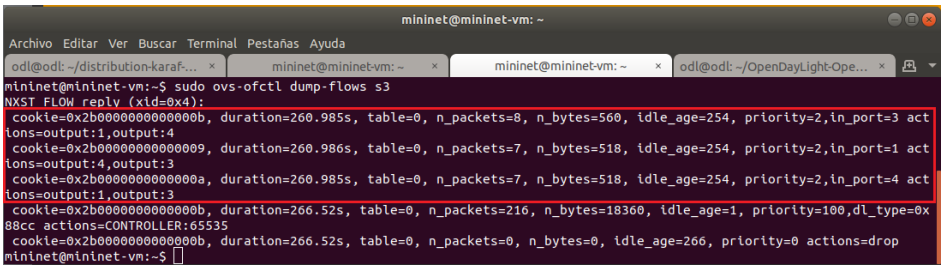

Figura 5.19: Flujos definidos inicialmente a remover y redefinir (recuadro rojo)

Utilizando OpenFlow Manager (OFM), se procede a revisar los flujos del recuadro y definir nuevas reglas para la conmutación de las rutas premium y estándar respectivamente (Figura 5.20)

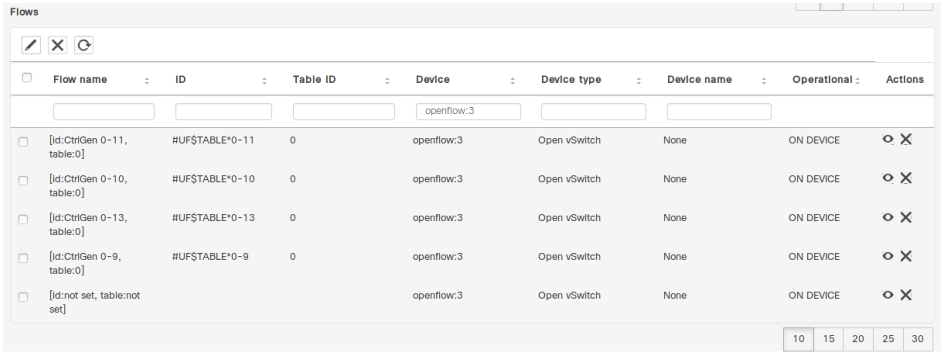

Figura 5.20: Detalle captura de pantalla pestaña de flujos en OFM

Como muestra el detalle de la Figura 5.20 y Figura 5.21, se puede visualizar el flujo definido o bien eliminarlo con los botones correspondientes.

Utilizando este último recurso, fácilmente podemos definir los nuevos flujos para las rutas premium y estándar. Para instalar las nuevas reglas de flujo, se debe tener el cuidado de hacerlo con estableciendo un valor mayor para el campo prioridad que el que se encuentra definido para las reglas que ya están instaladas, de esta manera no se afecta el comportamiento general de la red, pero establecemos un "overlay"(Figura 5.22), con las rutas de tráfico premium y estándar respectivamente.

Definidas las nuevas reglas, se procede a verificar los flujos instalados, a manera de ejemplo se

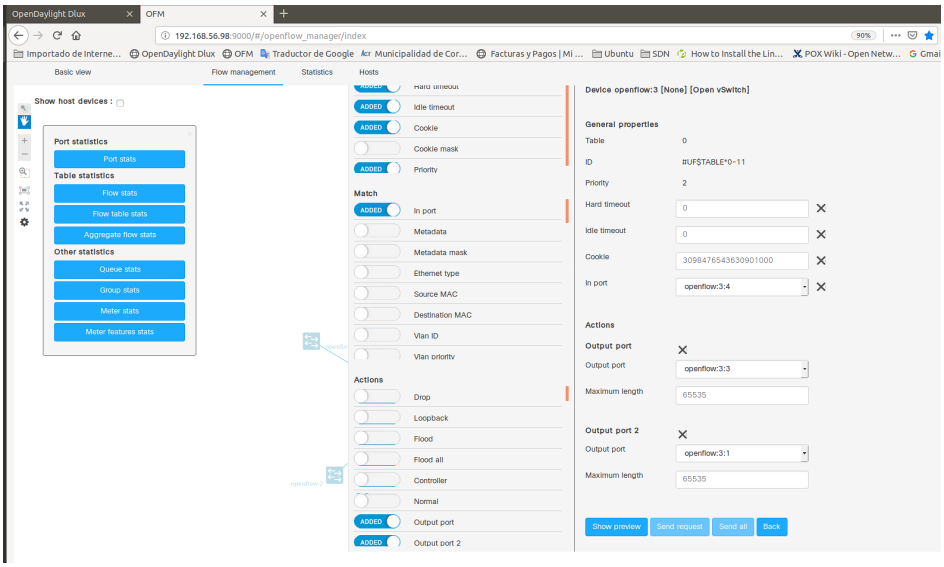

Figura 5.21: Captura de pantalla con detalle de flujo visualizado

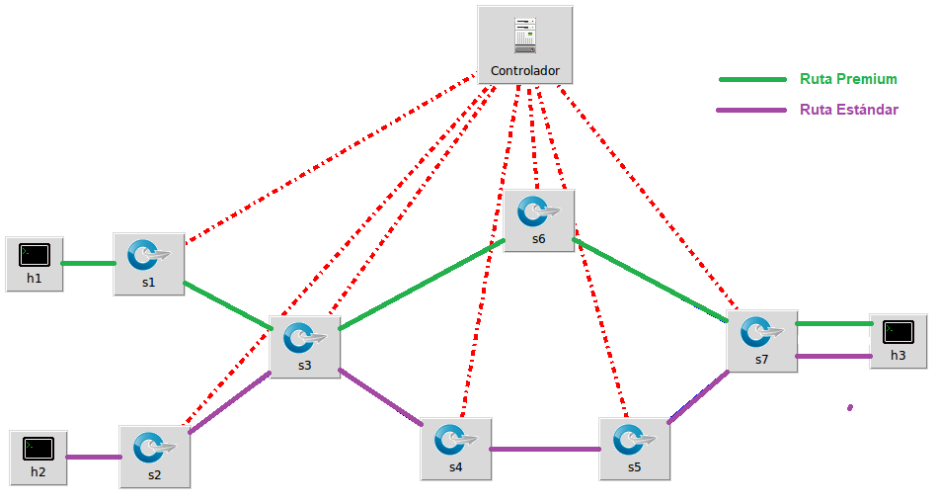

Figura 5.22: Rutas definidas en 'overlay'

muestra en los dispositivos openflow:3 y openflow:5, Figura 5.23 y Figura 5.24 respectivamente, se puede observar que las nueva reglas están en "overlay" sobre las anteriores, con una prioridad superior, que en este caso es 200, mientras que las reglas de flujo que tenía definida la red original, tienen un valor de prioridad de 2.

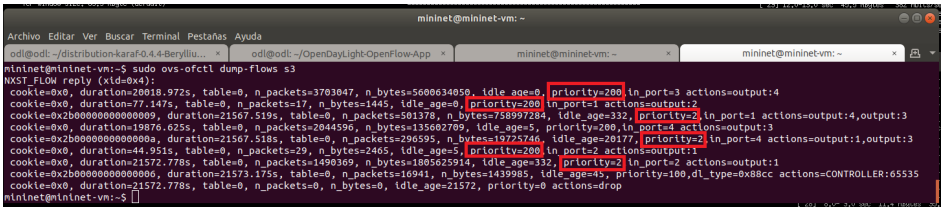

Figura 5.23: Flujos en el dispositivo openflow:3 después de agregar el 'overlay'

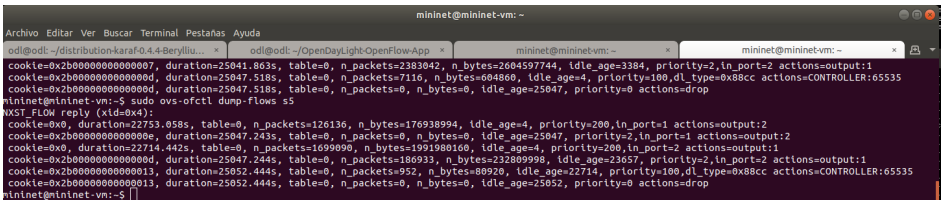

Figura 5.24: Flujos en el dispositivo openflow:5 después de agregar el 'overlay'

Una vez definidas todas las reglas de flujo para la red que llamamos "overlay", y que es la que tiene definidas las rutas para el tráfico premium y estándar, se procede a realizar una nueva prueba de performance.

Utilizando las terminales que teníamos abiertas en los hosts 1,2 y 3; y se repiten nuevamente varias pruebas simultaneas desde h1 y h2, con las mismas instancias cliente/servidor que teníamos iniciadas. De esta manera se obtienen nuevos valores de troughput. Desde la terminal del host 1 tendremos el troughput del tráfico premium y desde la terminal del host2 tendremos el troughput del tráfico estándar Figura 5.25.

En la Figura 5.25 se puede apreciar claramente, que el troughput es el esperado para cada tipo de tráfico. Esto es debido a que cada tipo de tráfico ocupa la ruta definida en la red "overlay', que asigna un camino con ancho de banda de 1Gbps para tráfico premium, y otro camino con un ancho de banda de 100 Mbps para el tráfico estándar.

Finalizando este apartado de prueba de performance, se muestra un comando del cli de Mininet que resulta muy cómodo y apropiado para este tipo de pruebas. Se trata de una versión de iperf incorporada al entorno Mininet y cuya sintaxis es simplemente:

- *iperf Nodo(x) Nodo(y)*

Donde x e y representan el nombre del nodo tal como aparece cuando se ejecuta el comando nodes.

Decimos que este comando resulta muy cómodo, pues no necesita otro argumento que los puntos de origen y destino entre los cuales se desea hacer una prueba de performance. A manera de ejemplo la Figura 5.26 muestra la salida del comando iperf, con la red sin las reglas de flujo del overlay (recuadro amarillo); y luego con las reglas de flujo de overlay aplicadas (recuadro rojo).

Como se puede apreciar en la Figura 5.26, el troughput entre los nodos h1–h3 y h2-h3, en las condiciones iniciales (recuadro amarillo; sin aplicar las reglas de flujo correspondientes), es muy similar en ambos casos; esto evidencia que todo el tráfico circula por la misma ruta. Sin embargo, luego de

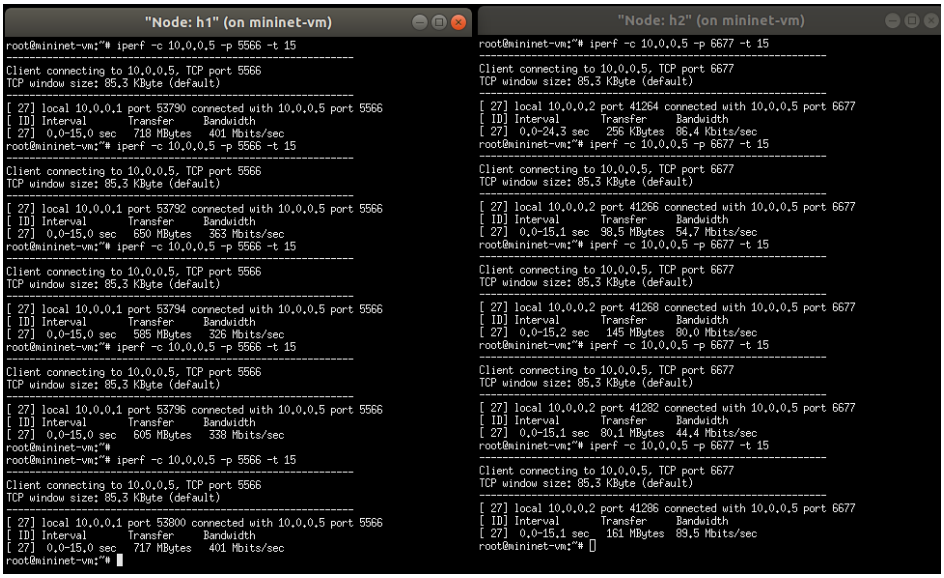

Figura 5.25: Pruebas de troughput para tráfico premium y estándar

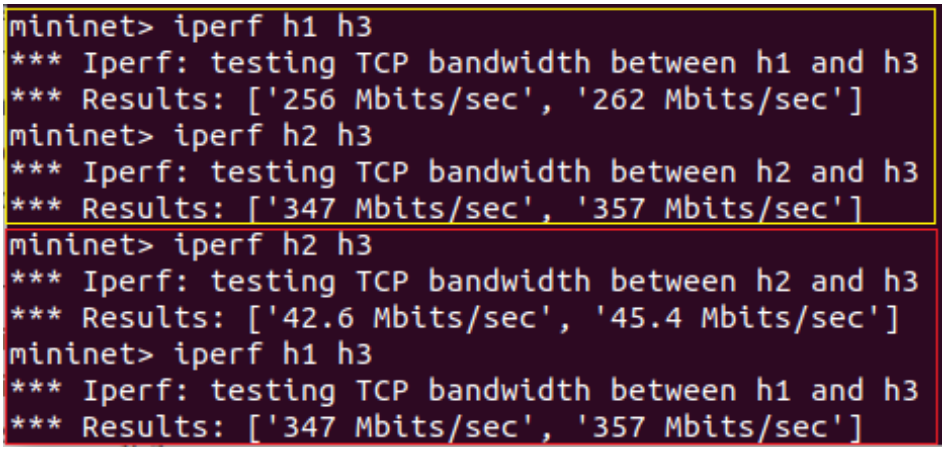

Figura 5.26: Comando iperf de Mininet sin reglas de overlay aplicadas y luego con regalas de overlay aplicadas

aplicar las reglas de flujo de nuestra red overlay, (recuadro rojo) el troughput de h1-h3 se acomoda al previsto para el tráfico premium, mientras que troughput h2-h3, se acomoda al previsto para el tráfico estándar; evidenciando que cada uno de los tráficos sigue la ruta definida para sus características de QoS.

5.3.2. Utilizando Técnicas de Modelamiento de Tráfico con SDN

Muchas veces es necesario limitar el tráfico saliente en una interfaz determinada, con el fin de administrar eficientemente los recursos de la red. Ante esta necesidad existen dos metodologías de limitación de ancho de banda: Policing y Modelamiento de Tráfico (Traffic Shaping). Mediante Policing se especifica la limitación a un máximo de tasa de transmisión para una clase de tráfico, si este umbral es excedido, una de las acciones inmediatas será ejecutada: transmitir, descartar, o remarcar. Con técnicas de admisión de Policing, los paquetes no se almacenan para posteriormente enviarlos, y en general, como consecuencia de esto, los paquetes que no cumplen con el criterio de admisión se descartan sin más.

Las técnicas de Modelamiento de Tráfico (Traffic Shaping) son más flexibles en el sentido en que operan. En vez de descartar el tráfico que excede cierta tasa determinada, reservan parte del tráfico sobrante mediante el uso de técnicas de encolamiento, con el fin de modelarlo a una tasa que la interfaz remota pueda manejar. El resto del tráfico excedente es inevitablemente descartado. Traffic Shaping (TS) es una técnica particularmente apropiada en situaciones en las cuales el tráfico saliente debe respetar una cierta tasa máxima de transmisión. Este proceso es realizado independientemente de la velocidad real del circuito.

Esto significa que es posible modelar tráficos de Web o Ftp a velocidades inferiores a las del receptor. TS puede hacer uso de las listas de acceso para clasificar el flujo y puede aplicar políticas restrictivas de TS a cada flujo. Policing descarta o remarca los paquetes en exceso si es que sobrepasan el límite definido. El tráfico que es originado en ráfagas se propaga por la red, no es suavizado como cuando se utilizan técnicas de Traffic Shaping, tal como se puede apreciar en la Figura 5.27. El Policing controla la tasa de salida mediante descarte de paquetes, por lo que disminuye el retardo por encolamiento.

Sin embargo, debido a estos descartes, el tamaño de la ventana deslizante de TCP debe reducirse, afectando el rendimiento global del flujo. En ciertos casos es necesario utilizar un camino con la velocidad adecuada para transmitir paquetes con alta o baja prioridad. Por ejemplo, si se tienen dos enlaces, uno con mayor velocidad que el otro, sería lógico plantear la metodología de transmisión de "Best Effort", para aquellos paquetes que tengan menor prioridad sobre el enlace de menor velocidad, tal como se vio en el "problema del pez", en el punto 5.1.1.

5.3.2.1. Traffic Shaping con Open sWitch

Como se menciona en la documentación referida a QoS de Open vWitch [109], un switch OVS admite la conformación del tráfico, utilizando Policing, para el tráfico que ingresa al switch, OVS y Traffic Shaping para el tráfico que egresa del switch.

Como se mencionó anteriormente, Policing es una forma simple de calidad de servicio que simplemente elimina los paquetes recibidos que exceden la tasa configurada, y debido a su simplicidad, el Policing es menos preciso y efectivo que el Traffic Shaping donde la configuración del tráfico de salida pone en cola los paquetes.

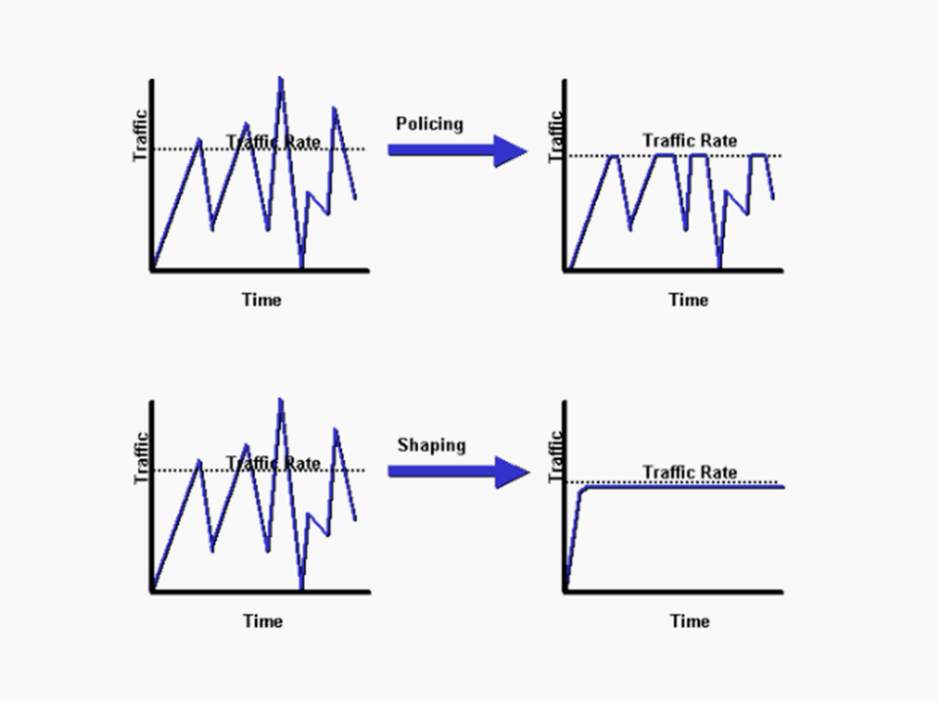

Figura 5.27: Comparativo de Policing vs. Traffic Shaping tomado de [16]

Es importante destacar que Open vSwitch no implementa QoS en sí. En cambio, puede configurar algunas, pero no todas, las características de QoS integradas en el kernel de Linux. Si necesita alguna característica de QoS que OVS no pueda configurar por sí mismo, entonces el primer paso es determinar si Linux QoS admite esa característica. Si lo hace, puede enviar un parche para admitir la configuración de Open vSwitch para esa función, o puede usar "tc"directamente para configurar la función en Linux. (Si Linux QoS no admite la función que desea, primero debe agregar esa compatibilidad a Linux ver [109], [110] y [111]).

5.3.2.2. Configurando colas en Open vSwitch

Si bien OpenFlow proporciona soporte limitado para QoS usando colas y medidores. La mayoría de los switchs de software admiten colas (en nuestro caso de estudio se utiliza OVS). Las colas se crean y configuran fuera del protocolo OpenFlow. OpenFlow proporciona la implementación al controlador para usar estas colas en los flujos.

Para implementar la calidad de servicio (QoS) en un switch OVS, necesitamos crear una política de QoS y colas con diferentes anchos de banda. Una vez hecho esto, necesitamos agregar estas colas en la política de QoS y conectar la política a un puerto el switch donde queremos implementar la QoS. Se muestra a continuación tres ejemplos de configuración de colas en OVS

***Ejemplo 1**: Creación de la política de QoS y dos colas en una interfaz (ver referencia [112])*

$ sudo ovs-vsctl set port "Nombre de la interface" qos=@newqos – –id=@newqos create qos type=linux-htb other-config:max-rate="X en bps" queues=1=@q1,2=@q2 – –id=@q1 create queue other-config:min-rate="X en bps" other-config:max-rate="X bps – –id=@q2 create queue other-config:min-rate="Y en bps" other-config:max-rate="Y en bps"

En el comando anterior, estamos creando una política de QoS de tipo linux-htb que tiene una

velocidad máxima de transferencia de datos de X expresado en bps, y la conectamos a la interface correspondiente (nombre de la interface que identificamos con el comando links). También estamos creando 2 colas q1 con id de cola = 1 y q2 con id de cola = 2 y agregamos estas colas en la política de QoS. q1 tiene una velocidad de transferencia máxima de X expresada en bps y una velocidad de transferencia mínima de Y expresada en bps. q2 tiene una velocidad de transferencia máxima de Y expresada en bps y una velocidad de transferencia mínima de Y expresada en bps. Una cosa a tener en cuenta es que el método QoS es solo de salida, lo que significa que estas velocidades de transferencia serán aplicables cuando los paquetes se envíen desde el puerto. Prestar atención en este ejemplo que la cola q1 tiene un tráfico sin restricciones pues X esta definido como valor máximo en "qos=@newqos – –id=@newqos create qos type=linux-htb other-config:max-rate=X" mientras que Y puede ser un valor menor para restringir tráfico saliente de esa interfaz a una tasa Y bps, definida en la cola q2.

***Ejemplo 2**: Creación de una política de QoS en el puerto s1-eth1con una tasa de 1 Gbps y una sola cola (q0) con una tasa de 4 Mbps.*

$ sudo ovs-vsctl – set port s1-eth1 qos=@newqos – –id=@newqos create qos type=linux-htb other-config:max-rate=1000000000 queues=0=@q0 – –id=@q0 create queue other-config:min-rate=4000000 other-config:max-rate=4000000

***Ejemplo 3**: Reserva de un ancho de banda de 3 Mbps en el puerto s1-eth4 de un switch, sobre un total de 7 Mbps (cola 0 por defecto)*

$ sudo ovs-vsctl – set port s1-eth4 qos=@newqos – –id=@newqos create qos type=linux-htb queues=0=@q0,1=@q1 – –id=@q0 create queue other-config:min-rate=0 other-config:max-rate=7000000 – –id=@q1 create queue other-config:min-rate=0 other-config:max-rate=3000000

Para finalizar la configuración, tenemos que mapear el ID (identificador) de cola requerido con un flujo en el switch. Si tomamos como referencia el ejemplo 3, el comando sería:

- *sudo ovs-ofctl add-flow s1 $in_port = 1, actions = enqueue : 4 : 1$*

A continuación, se muestra un ejemplo de implementación de colas para tráfico saliente mediante la utilización del entorno Mininet, un controlador OpenDayLight y switches OVS.

5.3.2.3. Descripción del escenario de prueba

Para la simulación de la técnica de QoS Traffic Shaping en un entorno SDN, se utiliza la topología que muestra la Figura 5.28, describe un escenario donde un dispositivo llamado s1, recibe todo el tráfico proveniente de afuera de la red. En el dispositivo s1, se configuran colas para atender flujos con distintos requerimientos, los cuales son atendidos por servidores que en la Figura 5 28 están representados como h2, h3, h4 y h5. El host h1 es desde donde se genera el tráfico para la simulación.

Los requerimientos son:

- *Tráfico para Host 2 un ancho de banda de 900 Mbps.*
- *Tráfico para Host 3 un ancho de banda de 100 Mbps.*
- *Tráfico para Host 4 un ancho de banda de 10 Mbps*
- *Tráfico para Host 5 un ancho de banda de 1 Mbps*

Para ilustrar brevemente otros recursos utilizados en estas pruebas se muestran capturas de pantalla realizadas en las interfaces gráficas del controlador OpenDayLight (ODL) y también de la aplicación OpenFlow Manager (OFM).

La Figura 5.29 muestra la topología de prueba en la interfaz gráfica Dlux del controlador OpnDayLight.

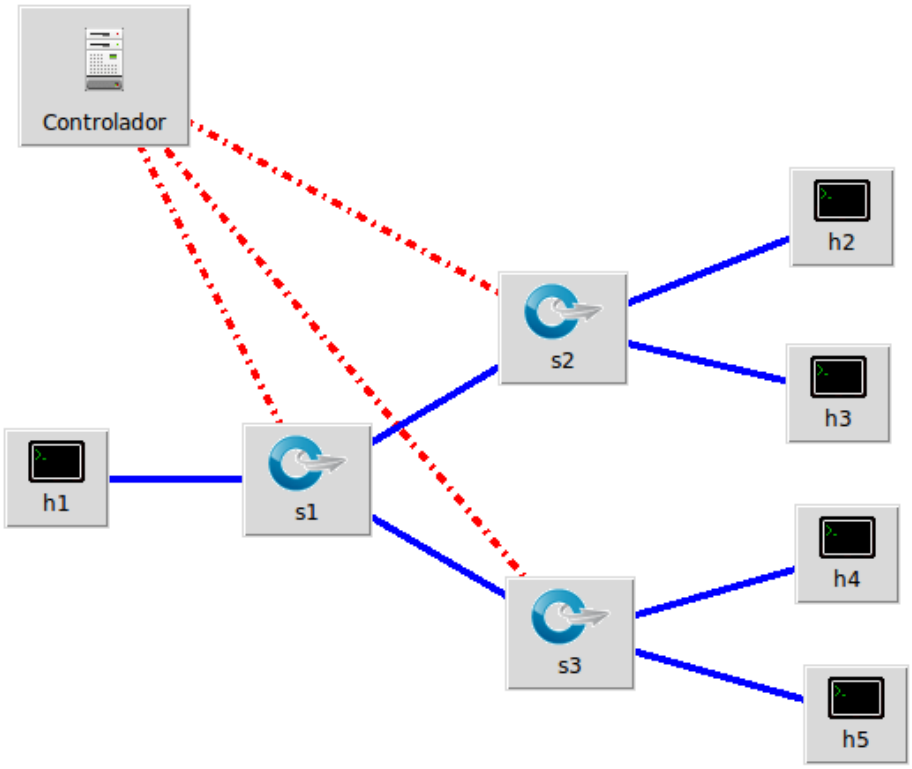

Figura 5.28: Topología de prueba para Traffic Shaping

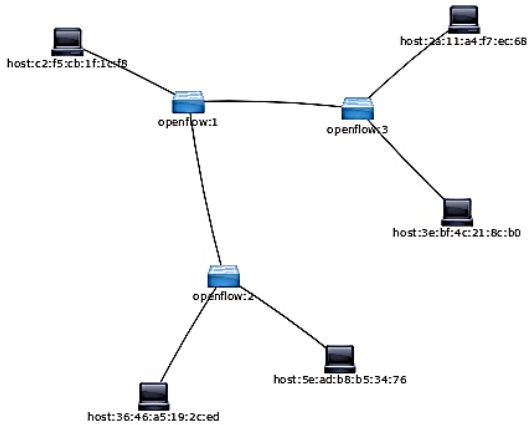

Figura 5.29: Topología de prueba en la interfaz Dlux de ODL

Para implementar QoS en la red utilizando mecanismos de Traffic Shaping, se implementan colas tipo Linux HTB (Hierarchical Token Bucket), ver referencias [109] [110] y [111], en las interfaces correspondientes a cada nodo. A continuación, se explica cómo usar las colas en OpenFlow 1.3 e implementar QoS en la topología de red de la Figura 5.28, con el controlador OpenDaylight (ODL).

Como se comentó en el punto 5.3.2.1, Open vSwitch no implementa QoS en sí mismo, la implementación se encuentra en el kernel de la MV anfitrión, por ende, antes de proceder a la configuración, verificamos la existencia de alguna configuración previa mediante los siguientes comandos:

- *sudo ovs-vsctl list qos*
- *sudo ovs-vsctl list queue*

Hecho esto, en caso de existir algo (se verifica la salida de los comandos anteriores), se procede a la eliminación de cualquier configuración previa mediante los comandos:

- *sudo ovs-vsctl –all destroy qos*
- *sudo ovs-vsctl –all destroy queue*

A continuación, se crea una política de QoS y se la agrega al puerto del switch correspondiente. Para aplicar la política en forma ordenada y sin riesgo de equivocar los puertos, se utiliza la salida del comando links del cli de Mininet (Figura 5.30), a los efectos de identificar perfectamente las interfaces donde se configuran las colas.

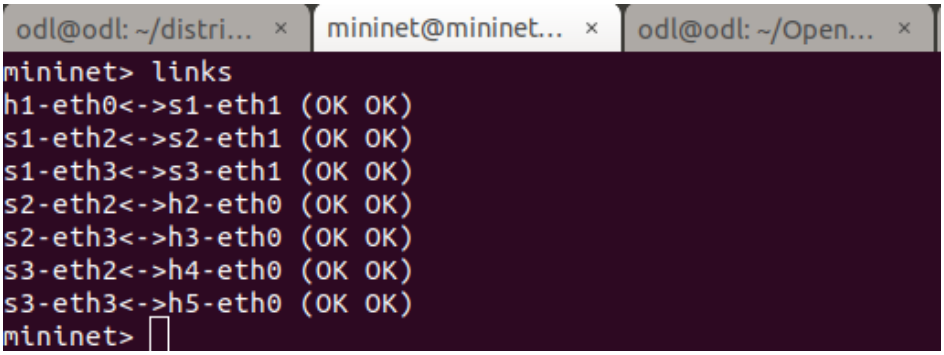

Figura 5.30: Comando links del cli de Mininet

Identificadas perfectamente las interfaces, se procede a la creación de las correspondientes colas para garantizar los requerimientos enunciados en el punto 5.3.1.1.

Creación de las colas:

Para el tráfico destinado al servidor h2 garantizamos una tasa de 900 Mbps:

- *$ sudo ovs-vsctl – set port s2-eth2 qos=@newqos – -id=@newqos create qos type=linux-htb other-config:max-rate=1000000000 queues=0=@q0 – -id=@q0 create queue other-config:min-rate=900000000 other-config:max-rate=900000000*

Para el tráfico destinado al servidor h3 garantizamos una tasa de 100 Mbps:

- *$ sudo ovs-vsctl – set port s2-eth3 qos=@newqos – -id=@newqos create qos type=linux-htb other-config:max-rate=1000000000 queues=0=@q0 – -id=@q0 create queue other-config:min-rate=100000000 other-config:max-rate=100000000*

Para el tráfico destinado al servidor h4 garantizamos una tasa de 10 Mbps:

- *$ sudo ovs-vsctl – set port s3-eth2 qos=@newqos – -id=@newqos create qos type=linux-htb other-config:max-rate=1000000000 queues=0=@q0 – -id=@q0 create queue other-config:min-rate=10000000 other-config:max-rate=10000000*

Para el tráfico destinado al servidor h5 garantizamos una tasa de 1 Mbps:

- *$ sudo ovs-vsctl – set port s3-eth3 qos=@newqos – -id=@newqos create qos type=linux-htb other-config:max-rate=1000000000 queues=0=@q0 – -id=@q0 create queue other-config:min-rate=1000000 other-config:max-rate=1000000*

Una vez que están creadas todas las colas en los dispositivos, se mapean a los flujos correspondientes al tráfico de cada uno de los servidores (h2, h3, h4 y h5), utilizando como "matching" las direcciones ip de origen y destino del tráfico. Para eso utilizamos Postman tal como se expuso en el punto 4.10. La Figura 5.31 muestra como ejemplo la asociación de la cola HTB de 1 Mbps a la salida de la interfaz s3-eth3 con ip de origen 10.0.0.1, e ip de destino 10.0.0.5.

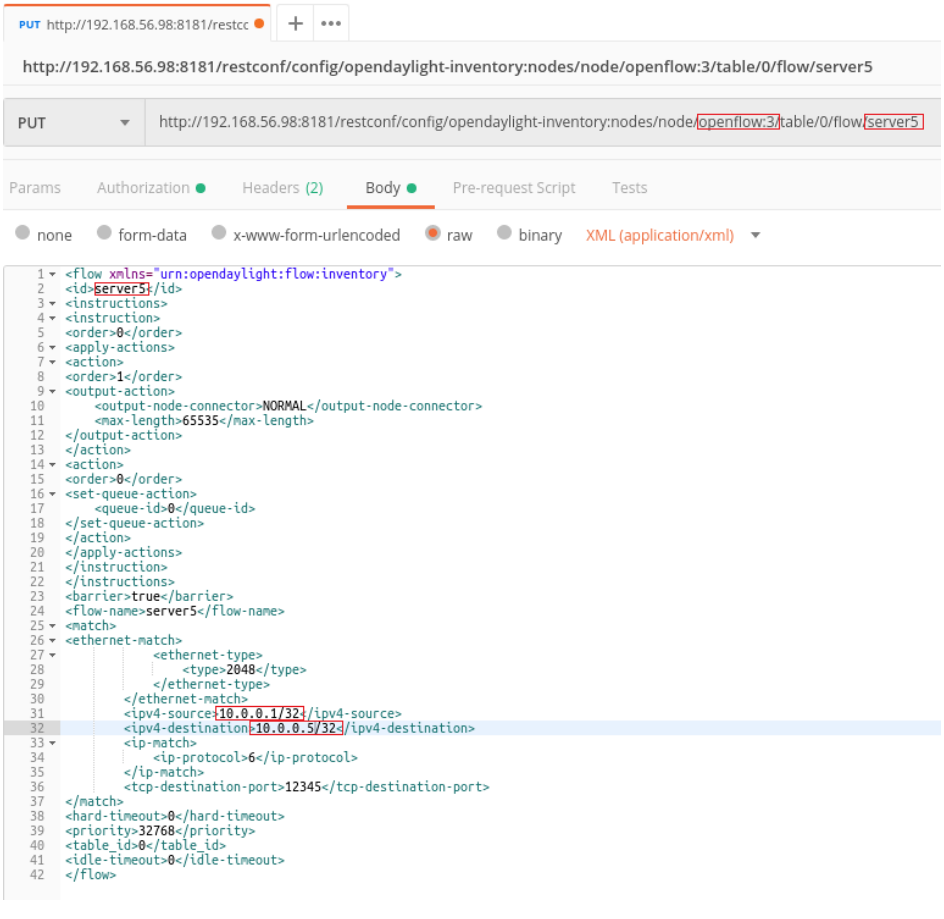

Figura 5.31: Asociación de cola con el flujo de tráfico hacia el servidor 5 con Postman

Comparativo de troughput en la topología de prueba sin QoS y con QoS

La Figura 5.32 muestra el troughput entre el host 1 y los respectivos servidores h2, h3, h4 y h5; sin la aplicación de ningún mecanismo de QoS

La Figura 5.33 muestra el troughput entre el host 1 y los respectivos servidores h2, h3, h4 y h5; con la aplicación de QoS

Los resultados de troughput sin QoS y con QoS son evidentes comparando los resultados que muestra la Figura 5.32 y la Figura 5.33, respectivamente, pudiendo observar que además obedecen a las tasas de velocidad definida oportunamente al configurar las respectivas colas.

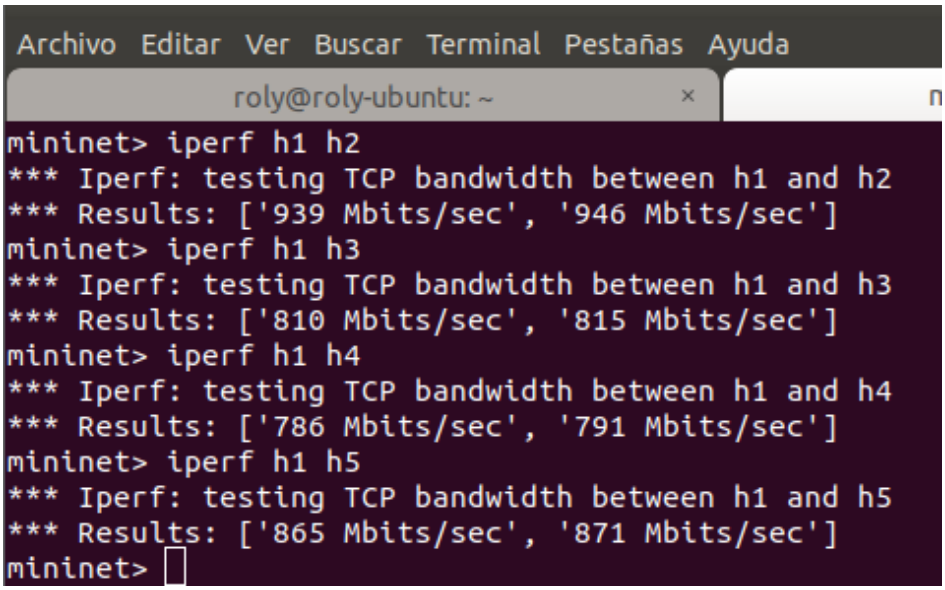

Figura 5.32: Troughput de la red sin mecanismos de QoS

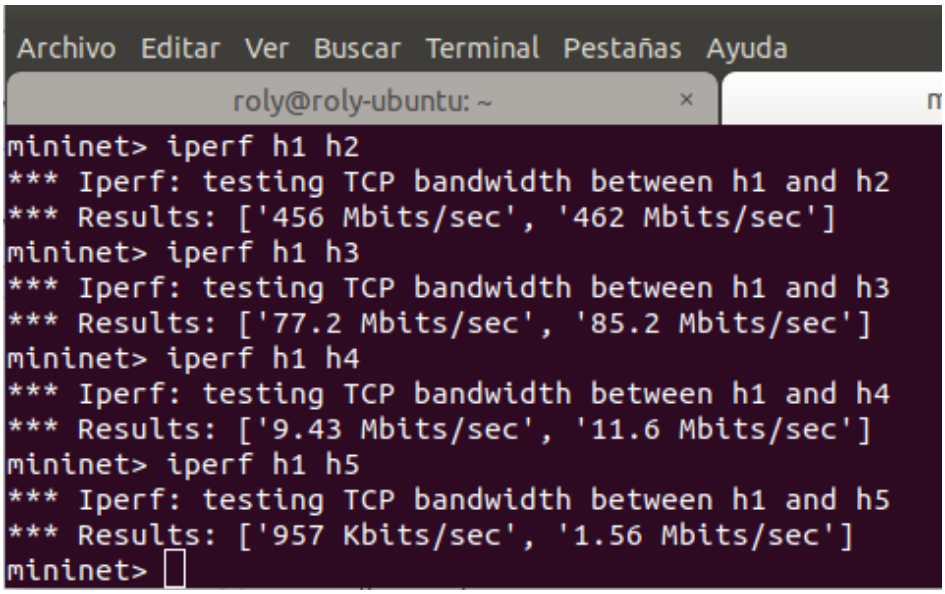

Figura 5.33: Troughput de la red con mecanismos de QoS aplicados

5.4. Unas palabras finales al capítulo 5

El objetivo previsto al utilizar ejemplos muy elementales es que permiten el abordaje de la temática con claridad conceptual. No obstante, la complejidad de escenarios y alcances que permite la tecnología SDN son enormes y representan un gran desafío que amerita la investigación estudio y aplicación de esta tecnología disruptiva por la separación del plano de control y posibilidades de interoperabilidad. Si bien en principio se pensó que SDN era superadora y reemplazaría a MPLS, rápidamente dio muestras de que no solo esto no es así sino que se complementan y potencian su campo de aplicación, incluso permitiendo escenarios y arquitecturas de red mucho más complejas, una muestra de esto se puede ver en el libro MPLS in the SDN Era de Antonio Sánchez-Monge y Krzysztof Grzegorz Szarkowicz [113], donde describen con claridad cuan importante resultó SDN como complemento de MPLS, dada la gran cantidad de fabricantes que desarrollaron características de MPLS en sus productos para satisfacer los requisitos de un mercado que cambia con gran velocidad y que requiere más que nunca adaptarse eficazmente a esos cambios.

Capítulo 6

Conclusiones y Recomendaciones

Cuando surgió la idea de plasmar nuestras experiencias en este proyecto de investigación en un libro, básicamente la propuesta fue escribirlo desde la visión del texto que hubiéramos deseado tener para afrontar inicialmente el estudio del tema. Y es con esta premisa que se estableció la secuencia de los capítulos y sus contenidos, mas allá que como dice su título se trata de las memorias de nuestro tránsito de aprendizaje de este nuevo paradigma que es SDN, comenzando desde la propia génesis de las tecnologías que lo sustentan y los estándares que lo rigen, para luego arribar a la puesta en marcha de practicas que complementen y permitan reproducir y afianzar dicho aprendizaje a quienes les resulte de interés su lectura.

6.1. Conclusiones

En particular, concluimos que el presente texto tiene un marcado carácter propedéutico, siendo a la vez lo suficientemente autocontenido para posibilitar un estudio inicial sin la necesidad de recurrir simultáneamente a múltiples fuentes para tal fin. Esto no lo afirmamos desde nuestro propio convencimiento, sino como resultado de las opiniones de quienes se tomaron el trabajo de leer y evaluar el texto previamente a su publicación, aún sin ningún conocimiento previo del paradigma SDN.

6.2. Recomendaciones

A manera de recomendación, nuestra propuesta inicial, es la reproducción de las experiencias simuladas analizando los resultados con los criterios expuestos, para luego ampliar o modificar los escenarios de simulación, para comparar los resultados obtenidos con los que se encuentran analizados en el libro.

6.3. Trabajos futuros

Es indudable que el paradigma SDN, modifica no solamente el diseño y gestión de redes de nueva generación, sino que está fuertemente ligado a nuevas áreas como, solo por citar algunas, Internet de las Cosas, Redes 5G o Gestión en Ciudades Inteligentes, con todo lo que esto implica. Por ende, nuestro grupo de investigación se encuentra en la tarea de llevar adelante un nuevo proyecto llamado "Análisis y Aplicaciones de Internet de las Cosas y Ciudades Inteligentes basadas Telecomunicaciones y Seguridad", donde esperamos aplicar lo aprendido y poder en el futuro próximo compartir el resultado del mencionado proyecto.

Bibliografía

[1] T. D. Nadeau y K. Gray, SDN: Software Defined Networks, *1ra ed. .º'Reilly Media, Inc.", 2013.*

[2] T.-Y. Liu, "The basic introduction of open vswitch," *2014. [En línea]. Disponible: https://es.slideshare.net/teyenliu/the-basic-introduction-of-open-vswitch*

[3] T. O. Foundation, "Openstack." *2018. [En línea]. Disponible: https://www.openstack.org/*

[4] D. Chowdhury, "Forces: An elastic routing architecture for nextgen sdn." ResearchGate, *08 2016.*

[5] S. Azodolmolky, Software defined networking with OpenFlow. *Packt Publishing Ltd, 2013.*

[6] O. N. Foundation, "Onf of-config.-1.2," *2014. [En línea]. Disponible: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow-config/of-config-1.2.pdf*

[7] ——, "openflow-spec-v1.3.1," *2012. [En línea]. Disponible: https://www.opennetworking.org/wp-content/uploads/2013/04/openflow-spec-v1.3.1.pdf*

[8] ——, "Sdn-architecture-overview-1.0," Open Networking Foundation, *12 2013.*

[9] ——, "Sdn architecture for transport networks," *2016. [En línea]. Disponible: https://www.opennetworking.org/wp-content/uploads/2014/10/SDN_Architecture_for_Transport_Networks_TR522.pdf*

[10] W. Stallings, "Software-defined networks and openflow," The Internet Protocol Journal, *vol. 16, num. 1, pp. 2–14, 2013.*

[11] O. N. Foundation, "Openflow switch specication version 1.5.1 (protocol version 0x06)," *2015.*

[12] S. Staff, "What is cisco opflex? - definition," *Jul 2014. [En línea]. Disponible: https://www.sdxcentral.com/cisco/datacenter/definitions/cisco-opflex/*

[13] Y. Stein y E. Haleplidis, "Sdn nfv, openflow and forces ietf-93," The Internet Engineering Task Force, *2007.*

[14] T. Lins, "Redes definidas por software (software defined networks) sdn," iMobilis, *2015.*

[15] opendaylight.org, "Opendaylight," *2016. [En línea]. Disponible: https://www.opendaylight.org/*

[16] C. T. Notes, "Comparing traffic policing and traffic shaping for bandwidth limiting," Cisco, *2004.*

[17] G. Pujolle, "Sdn (software-defined networking)," Software Networks: Virtualization, SDN, 5G and Security, vol. 1, pp. 15–48, 2015.

[18] J. Moy, "Ietf rfc 2328: Ospf version 2," 1998.

[19] K. Lougheed y Y. Rekhter, "Rfc 1267: Border gateway protocol 3 (bgp-3)," RFC, IETF, October, Tech. Rep., 1991.

[20] E. Rosen, A. Viswanathan, R. Callon y otros, "Multiprotocol label switching architecture," Internet Engineering Task Force, 2001.

[21] Microsoft, "Introducción a active directory," 2000. [En línea]. Disponible: https://support.microsoft.com/es-ar/help/196464.

[22] S. Shepler, B. Callaghan, D. Robinson, R. Thurlow, C. Beame, M. Eisler, y D. Noveck, "Rfc 3010: Nfs version 4 protocol," Internet Engineering Task Force (IETF), 2000.

[23] J. Sermersheim, "Rfc 4511-lightweight directory access protocol (ldap): The protocol," Internet Engineering Task Force (IETF), 2006.

[24] Wikipedia, "Modo de transferencia asincrona," 2019. [En línea]. Disponible: https://es.wikipedia.org/wiki/Modo_de_transferencia_as%C3%ADncrona

[25] D. Piscitello y J. Lawrence, "Rfc 1209, the transmission of ip datagrams over the smds service," SRI Network Information Center, 1991.

[26] Wikipedia, "High-level data link control," 2019. [En línea]. Disponible: https://es.wikipedia.org/wiki/High-Level_Data_Link_Control

[27] I. I. of Electrical y E. Engineers, "802.1d - mac bridges," 2006. [En línea]. Disponible: http://www.ieee802.org/1/pages/802.1D.html

[28] Wikipedia, "Broadcast, unknown-unicast and multicast traffic," 2019. [En línea]. Disponible: https://en.wikipedia.org/wiki/Broadcast,_unknown-unicast_and_multicast_traffic

[29] I. of Electrical y E. Engineers, "802.1aq - shortest path bridging," 2012. [En línea]. Disponible: http://www.ieee802.org/1/pages/802.1aq.html

[30] J. Touch y R. Perlman, "Transparent interconnection of lots of links (trill): Problem and applicability statement," Internet Engeneering Task Force (IETF), 2009.

[31] A. Sajassi, R. Aggarwal, A. Isaac, J. Uttaro, R. Shekhar, N. Bitar, y W. Henderickx, "Bgp mpls-based ethernet vpn," Internet Engineering Task Force, 2015.

[32] D. Farinacci, P. Traina, S. Hanks, y T. Li, "Generic routing encapsulation over ipv4 networks," Internet Engineering Task Force, 1994.

[33] D. Farinacci, D. Lewis, D. Meyer, y V. Fuller, "The locator/id separation protocol (lisp)," Internet Engineering Task Force, 2013.

[34] D. Meyer, L. Zhang, y K. Fall, "Rfc4984 report from the iab workshop on routing and addressing," Internet Engineering Task Force, 2007.

[35] *Wikipedia, "Multihoming," 2018. [En línea]. Disponible: https://es.wikipedia.org/wiki/Multihoming*

[36] *B. Furht y A. Escalante,* Handbook of cloud computing. *Springer, 2010, vol. 3.*

[37] *L. F. C. Project, "Production quality, multilayer open virtual switch,"* Open vSwitch, *2016.*

[38] *openvswitch.org, "ovs-vswitchd.8," 2018. [En línea]. Disponible: http://www.openvswitch.org/support/dist-docs/ovs-vswitchd.8.txt*

[39] *B. Claise, G. Sadasivan, V. Valluri, y M. Djernaes, "Rfc 3954: Cisco systems netflow services export version 9,"* Internet Engineering Task Force, *2004.*

[40] *P. Phaal, "sflow version 5,"* sFlow.org, *2004.*

[41] *B. Trammell, A. Wagner, y B. Claise, "Flow aggregation for the ip flow information export (ipfix) protocol,"* Internet Requests for Comments, RFC Editor, RFC 7015, *2013.*

[42] *C. S. Groups, "Span, rspan, erspan," Feb 2015. [En línea]. Disponible: https://learningnetwork.cisco.com/docs/DOC-26018*

[43] *C. System, "Basic command-line interface commands," 2014. [En línea]. Disponible: https://www.cisco.com/c/en/us/td/docs/ios/12_2/configfun/command/reference/ffun_r/frf001.pdf*

[44] *M. Seaman, "Link aggregation control protocol,"* ieee802.org, *vol. 1, p. 0399, 1999.*

[45] *I. I. of Electrical y E. Engineers, "802.1ag - connectivity fault management," 2007. [En línea]. Disponible: http://www.ieee802.org/1/pages/802.1ag.html*

[46] *O. N. Foundation, "Sdn technical specifications," 2018. [En línea]. Disponible: https://www.opennetworking.org/software-defined-standards/specifications/*

[47] *openvswitch.org, "Open vswitch how-to guides," 2016. [En línea]. Disponible: http://docs.openvswitch.org/en/latest/howto/*

[48] *A. Doria, F. Hellstrand, K. Sundell, y T. Worster, "General switch management protocol (gsmp) v3," 2002.*

[49] *C. M. University, "Clean slate architectures for network management," 2004. [En línea]. Disponible: http://www.cs.cmu.edu/~4D/*

[50] *T. Amin, "The nox controller," 2018. [En línea]. Disponible: https://github.com/noxrepo/nox*

[51] *R. Enns, M. Bjorklund, y J. Schoenwaelder, "Network configuration protocol (netconf),"* Internet Engineering Task Force, *2011.*

[52] *A. Doria, J. H. Salim, R. Haas, H. M. Khosravi, W. Wang, L. Dong, R. Gopal, y J. M. Halpern, "Forwarding and control element separation (forces) protocol specification."* RFC, *vol. 5810, 2010.*

[53] *S. C. S. Project, "Ethane: A security management architecture," 2006. [En línea]. Disponible: http://yuba.stanford.edu/ethane/*

[54] *O. N. Foundation, "Openflow switch specification version 1.0.0,"* Open Networking Foundation, *2009.*

[55] *——, "Software-defined networking: The new norm for networks,"* Open Networking Foundation, *04 2012.*

[56] *N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, y J. Turner, "Openflow: enabling innovation in campus networks,"* ACM SIGCOMM Computer Communication Review, *vol. 38, num. 2, 2008.*

[57] *S. University, "Stanford university," 2018. [En línea]. Disponible: https://www.stanford.edu/*

[58] *O. N. F. (ONF), "Onf," 2011. [En línea]. Disponible: https://www.opennetworking.org/*

[59] *O. N. Foundation, "Sdn architecture," 2016. [En línea]. Disponible: https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2014/10/TR-521_SDN_Architecture_issue_1.1.pdf*

[60] *——, "Software-defined networking (sdn) definition," 2008. [En línea]. Disponible: https://www.opennetworking.org/sdn-definition/*

[61] *——, "Onf miembros, socios, adherentes y colaboradores," 2018. [En línea]. Disponible: https://www.opennetworking.org/member-listing/*

[62] *H. Yin, H. Xie, T. Tsou, D. Lopez, P. Aranda, y R. Sidi, "Sdni: A message exchange protocol for software defined networks (sdns) across multiple domains,"* Internet Engineering Task Force (IETF), *2012.*

[63] *I. of Electrical y E. Engineers, "802.1w - rapid reconfiguration of spanning tree," Jul 2006. [En línea]. Disponible: http://www.ieee802.org/1/pages/802.1w.html*

[64] *J. Touch, M. Kojo, E. Lear, A. Mankin, K. Ono, M. Stiemerling, y L. Eggert, "Service name and transport protocol port number registry,"* The Internet Assigned Numbers Authority (IANA), *2013.*

[65] *iana.org, "Internet assigned numbers authority," 2018. [En línea]. Disponible: https://www.iana.org/*

[66] *M. Team, "mininet/mininet," 2017. [En línea]. Disponible: https://github.com/mininet/mininet/wiki/Mininet-2.2.2-Release-Notes*

[67] *——, "Mininet," 2018. [En línea]. Disponible: http://mininet.org/*

[68] *Wikipedia, "Redes definidas por software," 2018. [En línea]. Disponible: https://es.wikipedia.org/wiki/Redes_definidas_por_software*

[69] *A. O. Source, "Pox," 2018. [En línea]. Disponible: https://github.com/att/pox*

[70] *noxrepo, "Nox repo," 2018. [En línea]. Disponible: https://github.com/noxrepo/.*

[71] *O. Alliance, "What is osgi?" 2019. [En línea]. Disponible: https://www.osgi.org/developer/what-is-osgi/*

[72] B. Pfaff y B. Davie, "The open vswitch database management protocol," IETF Tools, 2013.

[73] J. Medved, L. Sedlak, M. Vitez, R. Varga, y T. Tkacik, "Yang tools:main," Jul 2013. [En línea]. Disponible: https://wiki.opendaylight.org/view/YANG_Tools:Main

[74] OpenContrail, "Opencontrail," 2017. [En línea]. Disponible: http://www.opencontrail.org/

[75] Juniper, "Juniper/contrail-controller," 2018. [En línea]. Disponible: https://github.com/Juniper/contrail-controller

[76] P. Floodlight, "Floodlight," 2018. [En línea]. Disponible: http://www.projectfloodlight.org/floodlight/

[77] S. Ryu, "Framework community: Ryu sdn framework," Ryu SDN Framework Community, 2015.

[78] Opennetworkinglab, "Flowvisor," 2013. [En línea]. Disponible: https://github.com/OPENNETWORKINGLAB/flowvisor/wiki

[79] V. Inc, "Vmware argentina: nube, movilidad, redes y seguridad," 2018. [En línea]. Disponible: https://www.vmware.com/ar.html

[80] vmware, "La virtualización y el centro de datos definido por software," Oct 2018. [En línea]. Disponible: https://www.vmware.com/ar/solutions/software-defined-datacenter/in-depth.html#networking.

[81] I. Juniper Networks, "Northstar controller," 2018. [En línea]. Disponible: https://www.juniper.net/us/en/products-services/sdn/northstar-network-controller/

[82] CISCO, "Orchestration and automation solutions," 2018. [En línea]. Disponible: https://www.cisco.com/c/en/us/solutions/service-provider/virtualization-automation.html#~stickynav=1

[83] H. P. D. Company, "Hp virtual application networks sdn controller," 2013.

[84] Wikipedia, "Transferencia de estado representacional," 2018. [En línea]. Disponible: https://es.wikipedia.org/wiki/Transferencia_de_Estado_Representacional

[85] opensource.org, "Open source initiative," 2018. [En línea]. Disponible: https://opensource.org/

[86] ——, "The open source initiative," 2018. [En línea]. Disponible: https://opensource.org/

[87] G. G. Public, "Gnu general public license, version 2," gnu.org, vol. 1991, 1989.

[88] T. F. S. F. (FSF), "Free software foundation," 2018. [En línea]. Disponible: https://www.fsf.org/

[89] C. Ltd, "Ubuntu releases," 2018. [En línea]. Disponible: http://releases.ubuntu.com/18.04/

[90] VirtualBox.org, "Virtualbox," 2018. [En línea]. Disponible: https://www.virtualbox.org/

[91] wireshark.org, "Wireshark," 2018. [En línea]. Disponible: https://www.wireshark.org/

[92] noxrepo, "Pox manual current documentation," 2015. [En línea]. Disponible: https://noxrepo.github.io/pox-doc/html/#

[93] openvswitch.org, "ovs-dpctl.8," Open vSwitch Manual, 2018.

[94] ——, *"ovs-ofctl.8,"* Open vSwitch Manual, *2018.*

[95] ——, *"ovs-vsctl.8,"* Open vSwitch Manual, *2018.*

[96] *C. Ltd., "Ubuntu," 2019. [En línea]. Disponible: https://ubuntu.com/#download*

[97] *I. SSH Communications Security, "Ssh (secure shell)," 2019. [En línea]. Disponible: https://www.ssh.com/ssh/*

[98] *nmap.org, "nmap.org," 2019. [En línea]. Disponible: https://nmap.org/*

[99] *CiscoDevNet, "Opendaylight openflow manager (ofm) app," Aug 2016. [En línea]. Disponible: https://github.com/CiscoDevNet/OpenDaylight-Openflow-App*

[100] *I. Postman, "The collaboration platform for api development," 2019. [En línea]. Disponible: https://www.getpostman.com/*

[101] *S. Keshav y S. Kesahv,* An engineering approach to computer networking: ATM networks, the Internet, and the telephone network. *Addison-Wesley Reading, 1997, vol. 1.*

[102] *M. Ye, H. Long, y G. Mirsky, "Ospf traffic engineering (ospf-te) link availability extension for links with variable discrete bandwidth,"* Internet Engineering Task Force (IETF), *2018.*

[103] *D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, y G. Swallow, "Rfc 3209: Rsvp-te: extensions to rsvp for lsp tunnels (2001),"* IETF Tools, *2001.*

[104] *L. Andersson, P. Doolan, N. Feldman, A. Fredette, y B. Thomas, "Rfc 3036: Ldp specification,"* Request for Comments, IETF, *2001.*

[105] *M. Dubuc y T. D. Nadeau, "Link management protocol (lmp) management information base (mib),"* Internet Engineering Task Force (IETF), *2006.*

[106] *P. Marques, T. Yamagata, K. Patel, K. Kumaki, y R. Raszuk, "Internal bgp as the provider/customer edge protocol for bgp/mpls ip virtual private networks (vpns),"* Internet Engineering Task Force (IETF), *2011.*

[107] *T. Bates, R. Chandra, D. Katz, y Y. Rekhter, "Rfc 4760: Multiprotocol extensions for bgp-4,"* IETF, *2007.*

[108] *J. Dugan, S. Elliott, B. A. Mah, J. Poskanzer, y K. Prabhu, "iperf-the ultimate speed test tool for tcp, udp and sctp,"* iPerf.fr, *2019.*

[109] *A. L. F. C. Project, "Quality of service (qos)," 2016. [En línea]. Disponible: http://docs.openvswitch.org/en/latest/faq/qos/#quality-of-service-qos*

[110] *M. A. Brown, "Classful queuing disciplines (qdiscs),"* Traffic Control HOWTO, *2006.*

[111] *bert hubert, "Linux tc(8)," 2019. [En línea]. Disponible: http://man7.org/linux/man-pages/man8/tc.8.html*

[112] *N. Kumar, "Qos using opendaylight," 2016. [En línea]. Disponible: https://www.talentica.com/blogs/qos-using-opendaylight/*

[113] *A. S. Monge y K. G. Szarkowicz,* MPLS in the SDN Era: Interoperable Scenarios to Make Networks Scale to New Services. *O'Reilly Media, Inc., 2015.*