UNIVERSIDAD DE  VALLADOLID

E.T.S.I. TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS ESPECÍFICAS DE TELECOMUNICACIÓN, MENCIÓN EN SISTEMAS DE TELECOMUNICACIONES

Aplicación de SDN en redes ópticas: análisis preliminar

AUTOR: RUBÉN DE PAZ VILLARROEL

TUTORES: RAMÓN J. DURÁN BARROSO
IGNACIO DE MIGUEL JIMÉNEZ

TÍTULO: **Aplicación de SDN en redes ópticas: análisis preliminar**

AUTOR: **Rubén de Paz Villarroel**

TUTORES: **Ramón J. Durán e Ignacio de Miguel**

DEPARTAMENTO: **Teoría de la Señal y Comunicaciones e Ingeniería Telemática.**

TRIBUNAL

Presidente: **Rubén M. Lorenzo Toledo**

Secretario **Ramón J. Durán Barroso**

Vocal: **Ignacio de Miguel Jiménez**

Suplente1: **J. Carlos Aguado Manzano**

Suplente 2: **Noemí Merayo Álvarez**

FECHA: **16 de Julio de 2015**

CALIFICACIÓN:

RESUMEN

La industria de las redes de comunicación ha llegado a un punto de inflexión debido al aumento de virtualización en el almacenamiento, la carga computacional y la llegada de la nube, que permite ofrecer servicios de computación a través de Internet. Esta situación hace que el sector tenga que tomar medidas al respecto ya que necesita reinventarse para poder ofrecer estos nuevos servicios emergentes.

Para conseguir estos retos, nace SDN (*Software Defined Networks* o Redes Definidas por Software) como una reacción de la industria a la necesidad de evolución que llevaba tiempo ocasionándose en la creación de redes. Su intención es sacar mejor partido a las aplicaciones e infraestructuras que hoy en día se gestionan de una forma ineficaz en algunos casos. El principio más básico detrás de SDN es la separación del plano de control y plano de datos. Por lo tanto, en las redes SDN la gestión de la red se realiza con un control centralizado para toda la red, y no manualmente en cada dispositivo como ocurre en las redes. En SDN, el apartado de control se le adjudica al controlador que es el cerebro de la red y el encargado de elegir la ruta por la que se enviaran los flujos.

El protocolo que se utilice para la comunicación entre los dispositivos de red y el controlador, es el "lenguaje" que deben de entender ambos para poder comunicarse entre ellos. Uno de los protocolos más extendidos al respecto es OpenFlow, que es utilizado principalmente por investigadores en redes experimentales.

El objetivo de este trabajo es hacer un estudio de las redes SDN y la evaluación sobre una posible inclusión de las mismas en redes ópticas. Esto permitirá estudiar el comportamiento de SDN y comprobar la eficiencia y utilidad de las mismas. Para ello, se explica principalmente el funcionamiento teórico de todos los aspectos relacionados con este tema.

PALABRAS CLAVE

SDN, OpenFlow, Redes Ópticas.

ABSTRACT

The network communication industry has to be reinvented if it wants to face the new challenges appeared with the arrival of cloud services and the increment in the virtualization of services such as computing and data storage.

SDN emerged as a solution to that problem as it allows to manage complex network infrastructures in an efficient way. The key concept of SDN networks is separation of the network control plane from the data plane. In contrast to traditional networks where the management of the devices is manually and independently performed, the management of network devices in SDN is performed by a controller who is the responsible of choosing the routing for each flow in the nework. The communications between the controller and the network devices is performed using a protocol.

One of the most extended protocols used for that aim is OpenFlow. The aim of this paper is to provide an introduction of SDN and OpenFlow and to analyze the different alternatives from the literature that allow the utilization of SDN concepts to control optical networks.

KEYWORDS

SDN, OpenFlow, Redes Ópticas.

AGRADECIMIENTOS

Me gustaría aprovechar esta oportunidad para mencionar a las personas sin las cuales, no hubiese sido posible llegar hasta este punto de mi carrera como estudiante. En primer lugar, quiero agradecérselo a mis compañeros de Universidad, a los que ya no considero como tal, sino como verdaderos amigos que he podido conocer gracias a esta carrera. Sin duda, nos hemos apoyado los unos a los otros en los buenos y sobre todo en los malos momentos y sin su apoyo a lo largo de estos cuatro años nada hubiese sido igual. Espero poder seguir contando con ellos y compartiendo muchos momentos de mi vida.

También quiero agradecérselo a mis amigos, ya que en todo momento me han ayudado a ver que las dificultades y problemas que me han ido surgiendo a lo largo de estos años, eran sólo eso y siempre tenían solución; gracias por hacerme ver la otra cara de la moneda.

Por supuesto, a mis padres y a mi familia, por haber depositado toda su confianza en mí desde el principio, y ser los principales motores durante toda mi vida.

Por último, agradecer a mis tutores a mis tutores Ignacio de Miguel y Ramón Durán porque su experiencia me ha servido para aprender en esta última etapa.

Estos agradecimientos no sólo están dedicados a la realización de este trabajo, sino a toda una trayectoria académica y personal, porque todos ellos me han hecho crecer como persona.

Muchas gracias

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo 1. INTRODUCCION

Cuando se crearon las comunicaciones a través de internet, el almacenamiento, los recursos de red y la computación se mantuvieron separados. Con el paso de los años y la demanda de carga computacional, fue cuando las organizaciones comenzaron a juntar estos elementos. Los *data centers*[1] se diseñaron originalmente para aplicaciones de red tradicionales, que tenían la finalidad de dar servicio al usuario, tales como servidores de correo o servidores de bases de datos. Más tarde se implantaron estos servicios de forma local con el propósito de facilitar la gestión y de que los usuarios pudieran comunicarse entre su propia empresa.

Hace unos 10 años surgió una transformación significativa. *VMware*[2] creó *hypervisor*[3] que permitía correr en un sistema operativo varios sistemas operativos cliente. En un principio, esta aplicación fue creada para ingenieros que necesitaban emular unas características específicas de un sistema operativo. Esto estuvo interesante porque se podía gestionar como un programa cualquiera y podía ser suspendido. El programa *hypervisor* tenía gran facilidad y flexibilidad para copiar máquinas virtuales, de esta forma el operador de red podía optimizar sus recursos ubicándolos en un data center. Esta nueva ubicación, ocasionó problemas de almacenamiento y refrigeración, porque con el tiempo se requería más poder de computación. Fue entonces cuando empresas como Amazon y Rackspace se dieron cuenta de que no estaban aprovechando todo su poder de computación y podían revendérselo a usuarios externos, lo que dio lugar a centros de datos multiusuario. Esto originó

---

1 Se denomina data center o centro de procesamiento de datos (CPD) a aquella ubicación donde se concentran los recursos necesarios para el procesamiento de la información de una organización.

2 *VMware* (*VM de Virtual Machine*) es una filial de EMC *Corporation* que proporciona software de virtualización disponible para ordenadores compatibles X86

3 Un hipervisor (en inglés *hypervisor*) o monitor de máquina virtual (*virtual machine monitor*) es una plataforma que permite aplicar diversas técnicas de control de virtualización para utilizar, al mismo tiempo, diferentes sistemas operativos.

otro problema que era el conflicto entre clientes de los *data centers*, ya que varios usuarios podían acceder al mismo sistema de almacenamiento. Para solucionar esto, la mayoría de los entornos asignaban direcciones IP[4] a todos sus dispositivos (físicos o virtuales) y todos podían acceder a sus recursos sin conflicto independientemente de su ubicación física.

Estos data centers podían ofrecer sus recursos de red aislados unos de otros, lo cual genera nuevos retos que no estaban presentes hasta el momento.

Los data centers fueron evolucionando, pero parecieron estancarse desde la llegada de IP, MPLS[5] y las comunicaciones móviles. Los routers y switches comerciales incluyen interfaces de gestión que permiten al operador de red configurarlos. Por ejemplo, los operadores de red pueden programar rutas estáticas, pero en última instancia las peticiones pasan por el sistema operativo. Esto no es un problema hasta que se desea experimentar con un nuevo protocolo y el *firmware* del dispositivo no lo permite. Así es como la distribución del plano de control llego a escena. Un dispositivo de red se compone de plano de control (que es el que gestiona la lógica de toda la red) y plano de datos (encargado del envío de paquetes). Si hay un controlador por cada dispositivo, se denomina control distribuido. Estos planos se comunican entre sí para crear las rutas de datos. Mientras que si un plano de control es centralizado, existiría un controlador que manejaría todos los dispositivos de red.

El concepto de plano centralizado forma el núcleo que lleva a lo que hoy conocemos como SDN. Aquí se vio un amplio mercado, en el que el poder de computación de un procesador podía gestionar un plano de control centralizado de varios dispositivos de red. [1]

---

4 Una dirección IP es una etiqueta numérica que identifica, de manera lógica y jerárquica, a una interfaz (elemento de comunicación/conexión) de un dispositivo (habitualmente una computadora) dentro de una red que utilice el protocolo IP (*internet protocol*), que corresponde al nivel de red del modelo OSI.

5 MPLS (siglas de *Multiprotocol Label Switching*) es un mecanismo de transporte de datos estándar creado por la IETF. Opera entre la capa de enlace de datos y la capa de red del modelo OSI.

Unos ingenieros de la universidad de Stanford crearon OpenFlow, un protocolo de estándares abiertos diseñado para este tipo de redes. OpenFlow interactúa entre el software de control de la red y los elementos de red, estableciendo mecanismos de entendimiento entre el flujo de datos y el plano de control. Este protocolo permite acceso directo a la manipulación del plano de reenvío de dispositivos de red tanto físico como virtual. [2]

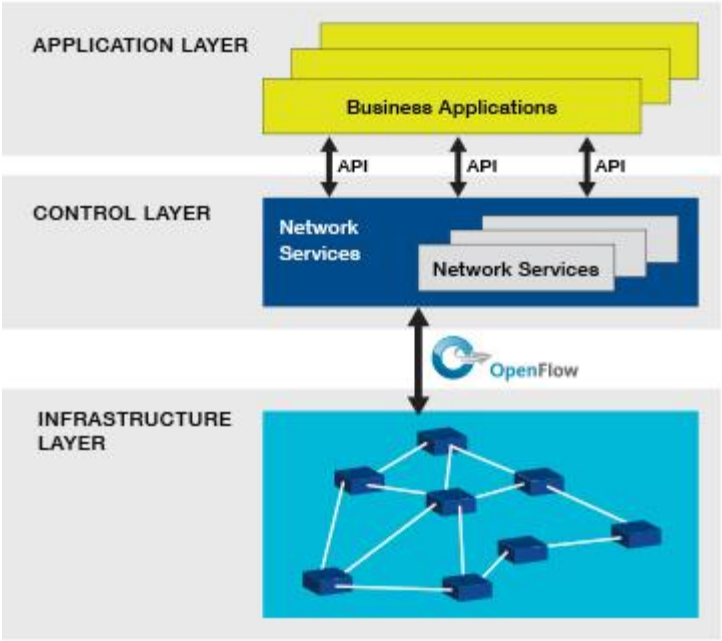

Figura 1: Protocolo OpenFLow, muestra gráficamente la diferenciación de las capas de aplicación, control e infraestructura.

Con el protocolo OpenFlow, una red puede ser gestionada como un solo elemento, no como un número de dispositivos individuales. El propio controlador es el que dice a los switches donde deben enviar los paquetes. Con esta tecnología, las decisiones que impliquen el movimiento de paquetes están centralizadas, por lo que la red puede ser programada independientemente de los switches.

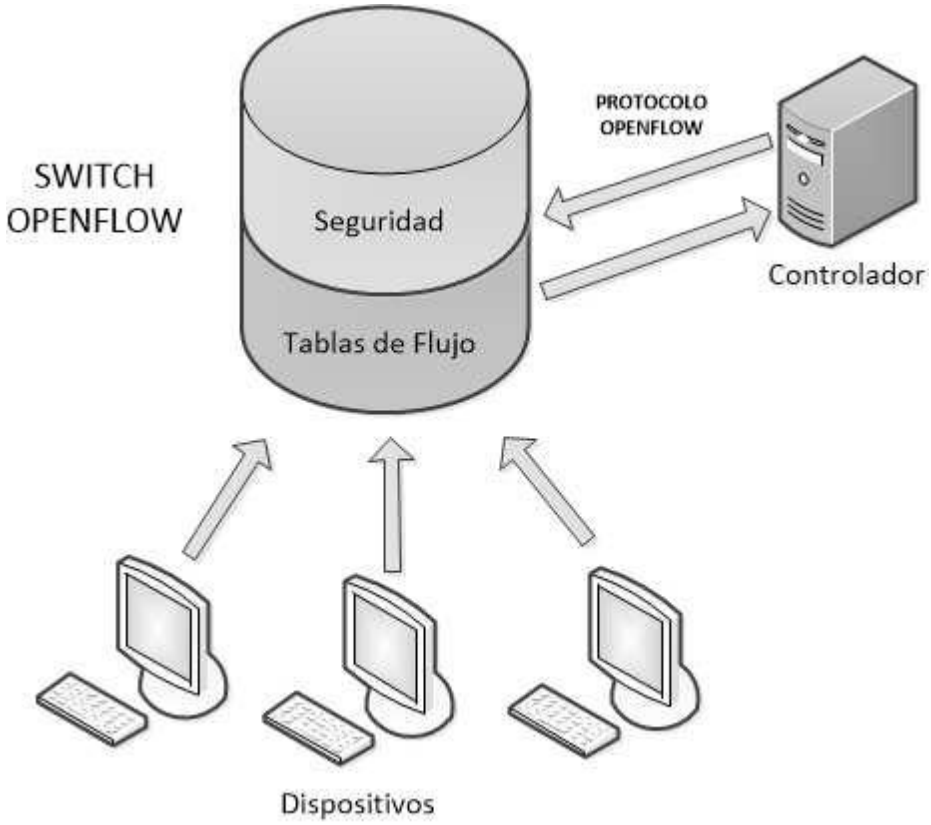

Figura 2: Switch OpenFlow

Los switches OpenFlow se pueden dividir en tres partes, tal cual a como se lo observa en la (figura 2): las tablas de flujo, la seguridad del canal y por último el protocolo OpenFlow. Las tablas de flujo indican a los dispositivos de red cómo tratar el paquete recibido. En el siguiente diagrama, se muestra cual es el tratamiento que sigue un paquete recibido por un switch OpenFlow. [3]

Figura 3: Diagrama de flujo del switch OpenFlow

Las pautas de reenvío se basan en el establecimiento de flujo. Un flujo consta de todos los paquetes que comparten una serie de características comunes. También, existe una gran variedad de parámetros que pueden especificarse para definir un flujo. Entre los posibles criterios el más claro es incluir los puertos por donde se reciben los paquetes cuando llegan, el puerto *Ethernet* de origen, la etiqueta VLAN[6], el destino *Ethernet* o el puerto IP y muchas más características que se pueden incluir en la definición de un flujo. Como se ve en la (figura 3) cuando un paquete que llega y no encuentra ninguna coincidencia con ninguna entrada de la tabla se debe crear un nuevo flujo. El paquete que no encuentre un flujo por el que enviarse, será enviado al controlador, entonces este define un nuevo flujo para el paquete y crea una entrada para la tabla. El

6 Una VLAN (acrónimo de *virtual* LAN, red de área local virtual) es un método para crear redes lógicas independientes dentro de una misma red física. Varias VLAN pueden coexistir en un único conmutador físico o en una única red física.

controlador envía la entrada al dispositivo de red por medio del protocolo OpenFlow para que sea añadida a las tablas de flujo ya existentes. Por último, el paquete se envía de vuelta al dispositivo de red para que este lo procese de manera adecuada con las nuevas entradas de la tabla.

Para comprobar el funcionamiento de todo esto, se puede emular la configuración personalizada de una red real utilizando distintos simuladores o emuladores como por ejemplo mininet. Este crea una red con dispositivos finales tales como un controlador, unos switches y unos hosts en un solo núcleo de Linux. Con la ejecución de un solo comando, se puede configurar como desees la simulación de la red, bien sea una de las predeterminadas que son más sencillas o una totalmente personalizada escrita previamente en un lenguaje de programación como Python[7]. Esta es la manera más rápida de hacer pruebas o investigación de redes SDN sin necesidad de crear una red física con los problemas que eso conlleva. Además es muy fácil interactuar con mininet mediante una API[8], ya que este está destinado a la investigación.

Una de las utilidades más importantes de SDN es su aplicación a redes ópticas. Este es un tema fundamental en el presente trabajo, ya que SDN combinada con las últimas tecnologías de transporte óptico, como las redes ópticas elásticas, permite a los operadores de red y proveedores de servicios de la nube personalizar su infraestructura de forma dinámica a las necesidades del usuario o de una aplicación. Por lo tanto, minimiza el capital adicional y los costos operativos necesarios al acoger nuevos servicios.

7 Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.

8 La interfaz de programación de aplicaciones (IPA), abreviada como API (del inglés: *Application Programming Interface*), es el conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usadas generalmente en las bibliotecas.

Capítulo 2. SDN

2.1. INTRODUCCION

En SDN la idea fundamental es dar todo el plano de control a un controlador software y no a uno hardware como venía siendo hasta ahora. Para llegar a esto un principio básico es separar el plano de control del plano de datos, lo cual se verá en el siguiente punto más detalladamente. Para ello el controlador debe llevar a cabo un control centralizado de la red y no un control distribuido por los dispositivos de red.

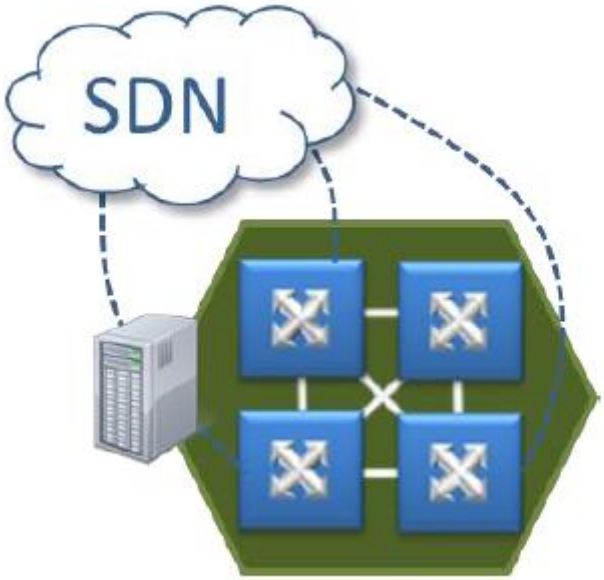

Figura 4: SDN

El encargado de la gestión de la red es el administrador de red, que maneja todo el tráfico mediante una aplicación de consola. Esto es más eficaz que lo que se venía haciendo hasta ahora, lo cual consistía en configurar individualmente los dispositivos, esa tardea resulta bastante ardua para grandes redes, por no hablar de la flexibilidad que nos permite hacerlo desde una aplicación. El administrador puede cambiar cualquier regla de los conmutadores de red cuando sea necesario dando o quitando prioridad, bloqueando tipos específicos de paquetes, o incluso introduciendo un tiempo determinado por el que ese paquete puede circular por un flujo. La finalidad de

todo esto es crear un flujo dinámico, capaz de responder a las necesidades del tráfico. [4]

En las redes tradicionales el reenvío del paquete depende de una programación previa, mientras que en SDN lo determina una interfaz de programación, así la forma de procesamiento de los paquetes ya no es estática. Se puede tratar a cada paquete por separado con los mensajes que se envían desde el software de control. Tradicionalmente se ha tratado esto de forma determinista, con el establecimiento de normas para todos los paquetes por igual. [3]

Según un informe de *Gartner*[9], en 2015 el 80% del tráfico en el data center será este-oeste, es decir, tráfico entre servidores en paralelo, no jerárquico. Las redes tradicionales no están preparadas para ese patrón de tráfico, porque son arquitecturas jerárquicas orientadas a tráfico norte-sur (cliente-servidor), en vertical. [5]

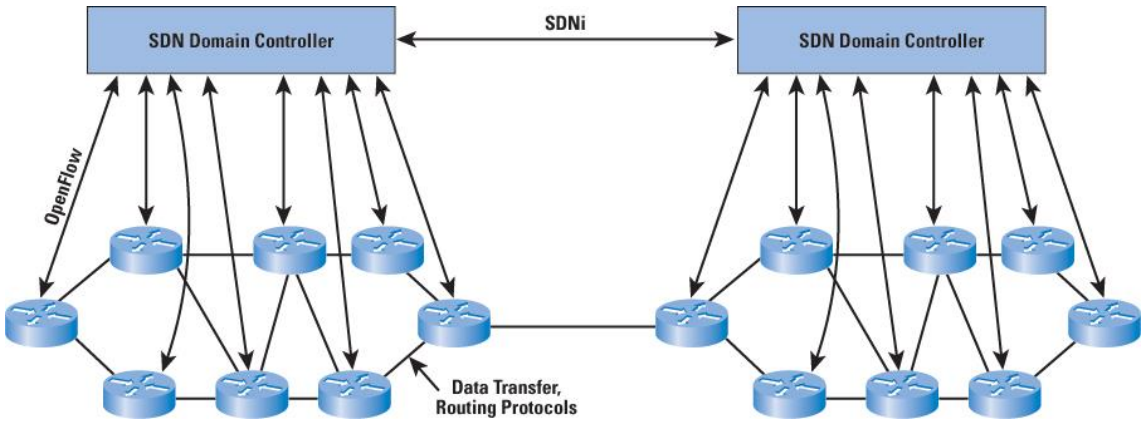

Figura 5: Imagen que muestra como puede ser el tráfico entre dispositivos

SDN busca la virtualización, y por ello, crea redes lógicas independientes de la infraestructura física que tienen por objetivo cumplir los exigentes requisitos de rendimiento, escalabilidad y agilidad necesarios en modelos de *cloud computing*[10].

9 *Gartner*, empresa consultora especializada en Tecnología Informática.

10 *Cloud computing* es un paradigma que permite ofrecer servicios de computación a través de Internet.

Los operadores de grandes data centers, los proveedores de cloud y grandes compañías ya están dirigiendo su futuro mirando a SDN, aunque todavía es temprano para que se generalicen. [5]

2.2. Separación del plano de datos y plano de control

El primer punto y más importante a tratar a la hora de hablar de SDN es la separación del plano de control y plano de datos. Esto no es una idea nueva sino que añade nuevos conceptos a las antiguas ideas que proponían esta separación, cómo de lejos tienen que estar estos dos planos y cuantas instancias se necesitan para satisfacer la alta disponibilidad y elasticidad necesarias.

Por un lado se puede hacer una lectura de borrón y cuenta nueva, eliminar lo que existe hasta ahora para implantar un control totalmente centralizado, aunque este es el caso más extremista. También se propone otra lectura con tendencia a evolucionar, que propone el control centralizado, pero que los dispositivos de red tengan una parte de "control distribuido" dentro de la misma idea. Esto se denomina una distribución hibrida. En el extremo opuesto a la idea original esta la forma clásica en la que el control está totalmente distribuido. En este modelo, cada dispositivo tiene su plano de control, que a su vez debe cooperar con otros planos de control. Este enfoque no evoluciona lo que ya existe ni presenta nada nuevo.

Para comenzar a explicar esta separación, se explicará el funcionamiento del plano de control. Este plano, a muy alto nivel establece un conjunto de datos local para crear la tabla de reenvío, que posteriormente, son utilizadas por el plano de datos para dirigir las entradas y salidas de un dispositivo. A este conjunto de datos se le denomina RIB (*Routing Information Base*). La RIB está en constante intercambio de información con otros planos de control de la red para mantenerse actualizada. También existe la FIB (*Forwarding Information Base*), pero lógicamente esta no se crea hasta que la RIB no es estable ya que depende de ella. [1]

La principal diferencia es que la RIB se utiliza en el plano de control, no se utiliza para el reenvío, mientras que la FIB se utiliza para el reenvío de paquetes.

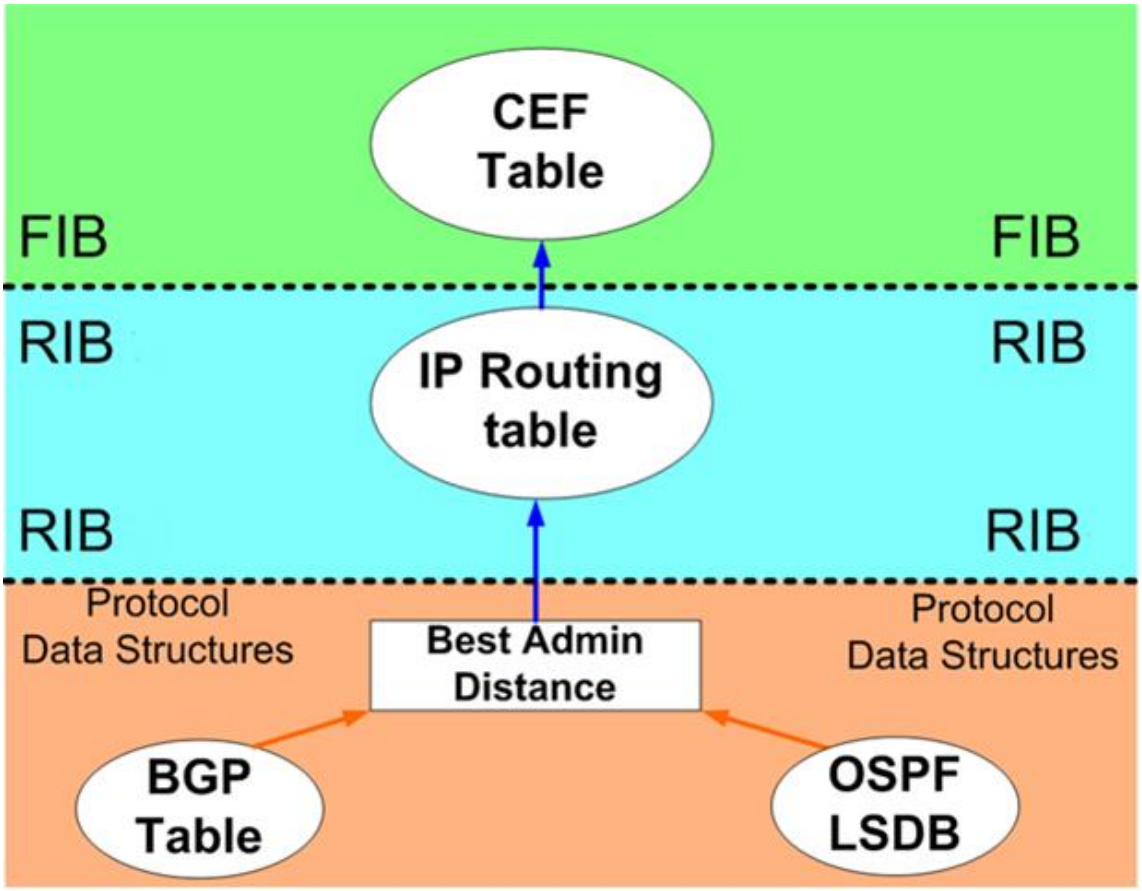

Figura 6: creación de las tablas FIB y RIB

En la figura 6 se muestra como desde el control del administrador se llega a construir la RIB, y a partir de esta la FIB.

El plano de datos, maneja los datagramas entrantes a través de una serie de comprobaciones a nivel de enlace que controlan que el dato este adecuado. Si un datagrama es correcto se mira la tabla FIB para ver cuál es el destino correspondiente a ese dato. Pero si el datagrama no concuerda con la tabla FIB, es desviado al controlador para que este decida qué hacer con el paquete.

Históricamente, las búsquedas en las tablas han demostrado mayor rendimiento en el reenvío de paquetes. Por lo tanto ha predominado este tipo de búsquedas, sobre todo para servicios de gran ancho de banda.

También, es importante hablar sobre la computación software y la computación hardware, ya que existen grandes diferencias entre ellos que intervienen en el funcionamiento de una red. Estas diferencias son: las características del destino, el rendimiento, la utilización del poder de computación, etc. Todas ellas dan lugar a que estos dos tipos de reenvío tengan diferentes características como velocidad de operación, memoria o diferente eficiencia a la hora de tratar paquetes de distintos tamaños. Estas diferencias, ocasionan una variedad de aspectos a tener en cuenta a la hora de decidir qué sistema de computación es adecuado para cada red, o si debe ser una mezcla entre ambos.

Además de las decisiones de reenvío, el plano de datos también demanda una serie de identificaciones o cabeceras. Gracias a ellas, se puede acortar el tiempo de búsqueda para el reenvío, identificando rápidamente cual es el destino del paquete. Existe un componente del plano de datos y del plano de control, como parte de la operación de reenvío el plano de datos, que tiene que tener una cierta cabecera en el datagrama para facilitar las operaciones. [1]

2.3. Limitaciones de las redes actuales

Debido al crecimiento de Internet, las empresas de equipos hardware comenzaron a implementar la lógica de reenvío de paquetes en hardware (plano de datos), separada del plano de control y los ISP[11] para poder gestionar sus redes crecientes, y así poder aportar a sus clientes servicios que las hiciesen más seguras (como VPN).

11 El proveedor de servicios de Internet (ISP, por la sigla en inglés *de Internet Service Provider*) es la empresa que brinda conexión a Internet a sus clientes. Un ISP conecta a sus usuarios a Internet a través de diferentes tecnologías

Para hacer frente a las necesidades actuales de empresas y usuarios finales, las redes deben dejar a un lado algunas características que las limitaban y que con SDN mejoran considerablemente. Algunas de estas limitaciones son:

- Dependencia del fabricante: el fabricante de los dispositivos de red determinaba la forma de crecer de una red, ya que las redes se ven frenadas por los ciclos de producción de los equipamientos por parte de los vendedores. Estos ciclos pueden abarcar varios años.
- Complejidad: Las redes tradicionales son mucho más difíciles de gestionar. Los dispositivos se tienen que configurar de forma individual y en una red amplia, eso es una tarea muy ardua.
- Mala adaptación al cambio: son redes muy estáticas y lleva grades procesos el realizar cambios en ellas.
- Políticas inconsistentes: Se quieren unificar todas las políticas de envío. Para ello los administradores de red, cada vez que tienen que configurar un dispositivo, pueden tardar horas hasta que se reconfigura ACL[12] en toda la red.
- Imposibilidad de escalabilidad: La red no crece de la misma forma que lo hacen las necesidades que requieren los clientes. Sin embargo, la red se vuelve más compleja con la suma de cientos de miles de aparatos de red, que deben ser configurados y gestionados.
- Mala flexibilidad: si se quiere experimentar con nuevas aplicaciones, SDN ofrece mejores configuraciones porque se controla con un solo dispositivo.
- Alto coste: Garantizar la calidad de las redes demanda una gran inversión en equipos con un *firmware* propietario. De esto se pasaría a dispositivos genéricos que interpretan OpenFlow y gestionan su tabla de flujos.

12 Una lista de control de acceso o ACL (del inglés, *Access Control List*) es un concepto de seguridad informática usado para fomentar la separación de privilegios.

SDN es la posibilidad soñada por los administradores de sistemas y redes. Porque es capaz de reaccionar bajo demanda a las peticiones de las aplicaciones.

2.4. Características de las SDN

Se propone centralizar el control de la red SDN con el controlador. La utilización del protocolo OpenFlow, con el objetivo de estandarizar los dispositivos, ayuda a que se puedan dar una serie de rasgos representativos de las redes SDN.

Las características que se derivan de estas particularidades de SDN, son mejoras que se venían requiriendo con respecto a las arquitecturas de red tradicionales:

- Se puede desplegar en todas partes: Puede controlar la red en función a los sistemas informáticos conectados al controlador.
- Altamente Programable: El control de la red es directamente programable. Se pueden configurar las rutas por las que circula el tráfico de datos. Esto es muy útil en casos en los que se congestiona un camino. [6]
- Cualquier desarrollador puede desarrollar software para la red. Las políticas son de código abierto, esto aporta una flexibilidad.
- Ágil: permite al administrador de red hacer cambios o modificaciones de forma dinámica, gracias a tener un monitoreo constante de la red. Además pueden ser escritos por ellos mismos porque los programas son de código abierto.
- Gestión centralizada: La inteligencia de la red está centralizada en un software llamado controlador SDN. Toda la red se gestiona a partir de este punto que aplica las órdenes necesarias separar y dirigir el tráfico de red.
- Control del tráfico de la red: se pueden asignar los recursos de la red de forma dinámica. Por ejemplo SDN puede repartir el tráfico de los

picos entre varios servidores sin necesidad de tener que crear un sistema que sea capaz de soportar estos picos.

- SDN puede disminuir los gastos operativos, el número de errores y el tiempo de inactividad de la red. Todo esto permite la configuración automática de la red y reduce la configuración manual. [7]

2.5. Arquitectura SDN

La arquitectura SDN se basa en un servidor controlador, el cual inspecciona los patrones de flujo de datos de acuerdo con las reglas que tenga definidas, y los dispositivos de red, estos se encargan de recibir las órdenes del controlador y aplicarlas.

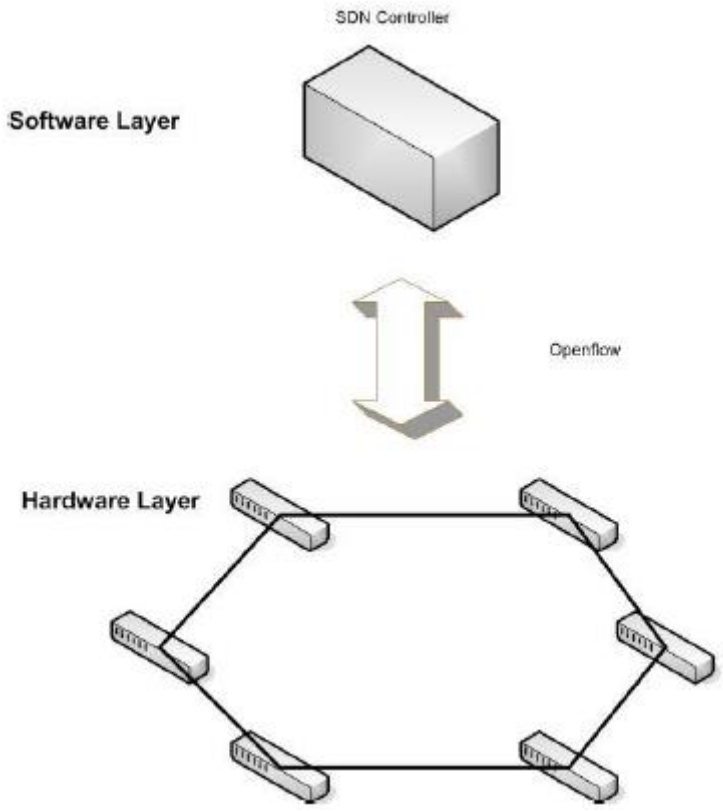

Figura 7: Arquitectura SDN

En la figura 7 se puede observar como el controlador gestiona toda la red, comunicándose con los switches, mediante el protocolo OpenFlow, para establecer los patrones de flujo.

La representación de una arquitectura SDN se ve en la siguiente figura 10. A continuación, se explican las diferentes en las que se divide.

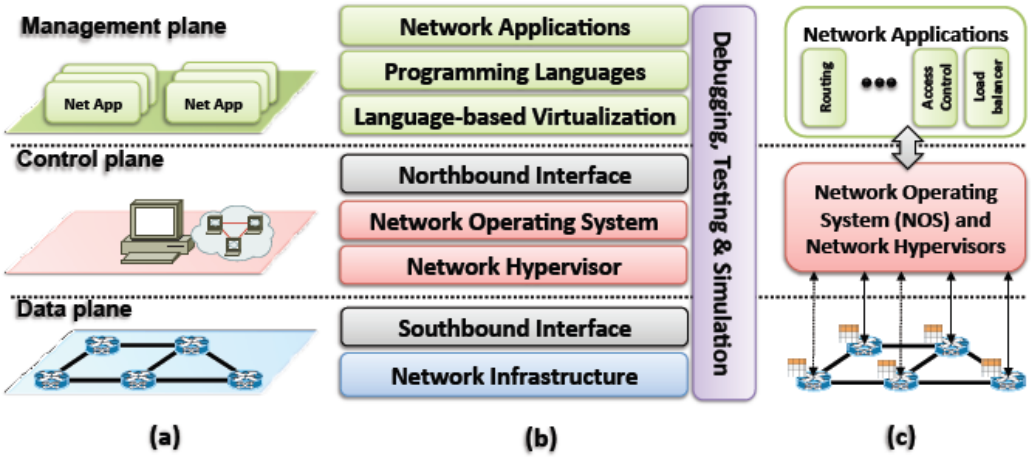

Figura 8: Redes SDN en a) Planos, b) Capas y c) sistema de diseño de la arquitectura.

2.5.1. 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. La inteligencia de la red se transfiere lógicamente al controlador SDN, centralizado como se muestra en la figura 10 (a). Es muy importante también que estas interfaces abiertas permitan al controlador programar dinámicamente los dispositivos, algo que en las redes tradicionales no se podía hacer.

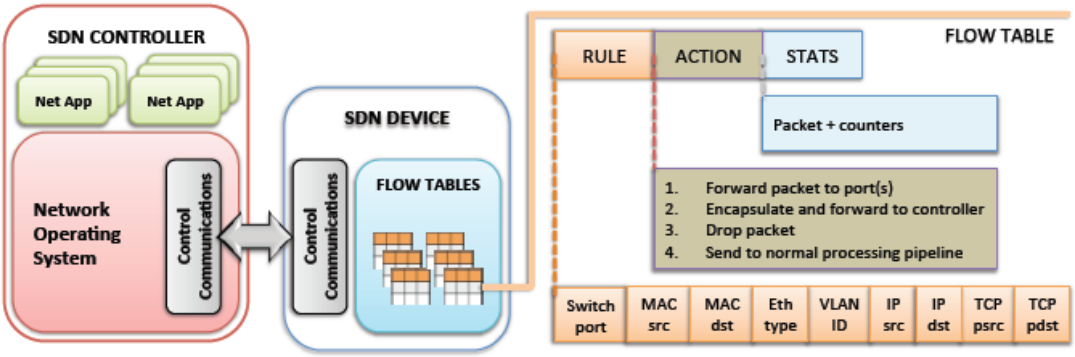

Figura 9: Dispositivos SDN OpenFlow.

En SDN hay dos elementos principales: el controlador y los dispositivos de red, como muestra la figura 11 : los dispositivos de red están especializados en el reenvío de datos, mientras que el controlador es una pila de software que se ejecuta en una plataforma hardware. Los dispositivos de red OpenFlow están basados en las tablas de flujo, cada entrada de la tabla tiene tres partes:

- Las reglas de concordancia.
- Las acciones que se realizan en los paquetes que cumplen las reglas anteriores.
- Contadores que mantienen una lista de estadísticas de los paquetes que concuerdan.

En un dispositivo OpenFlow, una secuencia de tablas de flujo define como se van a tratar los paquetes, así cuando llega un nuevo paquete se inicia un proceso de búsqueda en la primera tabla y termina encontrando su regla de envío o sin coincidencias. Es entonces cuando el paquete se envía al controlador y éste vuelve con una regla para saber cómo tratarlo. Las acciones que se realizan con un paquete son las siguientes:

1. Enviarlo al puerto o puertos de salida.
2. Encapsularlo y enviarlo al controlador.
3. Descartarlo.
4. Enviarlo por el procedimiento normal.
5. Enviarlo a siguientes tablas de medición, posiblemente introducidas por el protocolo OpenFlow.

Hay diversos tipos de dispositivos OpenFlow en el mercado; tanto productos comerciales, como de código abierto. La mayoría de los switches, de este tipo, que hay en el mercado tienen poca memoria, pero esto está cambiando y cada vez incorporan más, esto demuestra que las tablas de flujo están creciendo a un ritmo muy alto para satisfacer las necesidades de las redes futuras. Hay desde dispositivos para pequeñas empresas (switch GbE), hasta dispositivos

de grandes centros de datos con 100GbE y una velocidad de conmutación de Tbps.

2.5.2. *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. Sin embargo, estas API están todavía muy enlazadas a dispositivos físicos de reenvío de paquetes en una infraestructura física.

Un switch puede tardar hasta dos años de desarrollo para que esté listo en el mercado. Contando con el desarrollo de software, las API *southbound* pueden ser una barrera para la introducción y aceptación de estos dispositivos. Por esta razón ha sido tan bien recibido OpenFlow, ya que promueve la interoperabilidad, lo que incentiva el desarrollo de dispositivos de red por distintos proveedores.

Hasta el momento OpenFlow es el estándar más aceptado e implementado. Ofrece tres fuentes de información a los sistemas operativos:

1. Los dispositivos que envían mensajes basados en eventos al controlador cuando hay algún cambio.
2. Las estadísticas del flujo que generan los dispositivos de red y son recogidas por el controlador.
3. Los mensajes en paquetes que se envían al controlador cuando los dispositivos no saben qué hacer con el flujo entrante, o porque se especifica que ese mensaje se debe enviar al controlador.

Estos son los datos esenciales para proporcionar información al nivel de flujo al sistema operativo de red.

2.5.3. Hipervisores de red

La virtualización ya es un hecho en los tiempos que corren. Según informes recientes, hay más servidores virtuales que servidores físicos.

Los hipervisores permiten a las máquinas virtuales compartir recursos 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. Una característica muy importante es que las máquinas virtuales se pueden migrar fácilmente de un servidor físico a otro. A pesar de todo esto la red sigue siendo estáticamente configurada punto a punto.

La infraestructura de red debería ser capaz de soportar topologías de red arbitrarias y esquemas de direccionamiento. Así, cada cliente tiene que ser capaz de configurar nodos de computación y red simultáneamente. Esto se cree poder hacer con tecnologías como VLAN, NAT[13] y MPLS. El problema de estas tecnologías es que están ancladas en el pasado en una configuración estática y no pueden configurar la red de manera global. Así el aprovisionamiento de la red puede tardar meses, mientras que el cálculo de aprovisionamiento se hace en minutos. Existe esperanza de que esto pueda cambiar con SDN y las nuevas técnicas de *tunneling*[14] como VXLAN[15].

FlowVisor es una de las primeras tecnologías para virtualizar un SDN. Su idea básica es permitir a múltiples redes lógicas compartir la misma infraestructura de redes OpenFlow. También se utilizan *OpenVirteX* y *AutoSlice*.

2.5.4. Sistema operativo de red / controlador

SDN se comprometió a simplificar la gestión de red por medio del control de lógica centralizada, que ofrece un sistema operativo de red (NOS[16] o controlador). Con NOS, los desarrolladores ya no tienen que preocuparse de

13 La traducción de direcciones de red o NAT (del inglés *Network Address Translation*) es un mecanismo utilizado por routers IP para intercambiar paquetes entre dos redes que asignan mutuamente direcciones incompatibles.

14 Técnica que consiste en encapsular un protocolo de red sobre otro

15 VXLAN: un marco para la superposición de la virtualización de la capa 2 de red sobre la capa 3.

16 Sistema operativo de red, *Network Operating System*.

los detalles de nivel bajo de distribución de datos entre los elementos de enrutamiento. Por ejemplo, estos sistemas pueden crear un nuevo entorno capaz de impulsar la innovación a un ritmo más rápido reduciendo la complejidad en la creación de nuevos protocolos. El controlador, como es natural, es un elemento esencial en la lógica de control para generar la configuración de red en función de las políticas definidas por el operador de red.

2.5.5. *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.

También es esencial que estas interfaces sean abiertas para promover la portabilidad de aplicaciones. Una API hacia el norte, es comparable a POSIX[17] en los sistemas operativos.

En este tema de interfaces hacia el norte cada controlador proporciona y define la suya. Los lenguajes de programación proporcionan mecanismos como la composición de aplicaciones, un plano de datos transparente de tolerancia a fallos y bloques de construcción para facilitar el módulo de software y el desarrollo de aplicaciones de gran alcance.

Es muy difícil que una única interfaz hacia el norte se estandarice. Esto se debe a que los requisitos para las diversas aplicaciones de red distintas, tal como las aplicaciones de seguridad son muy diferentes de las aplicaciones financieras o de enrutamiento.

17 POSIX es el acrónimo de *Portable Operating System Interface*, y X viene de UNIX como seña de identidad de la API.

2.5.6. La virtualización basada en el Lenguaje

Las soluciones de virtualización poseen dos características indispensables y es que son capaces de expresar modularidad y de permitir diferentes niveles de abstracciones. Esto, sin dejar de garantizar las características deseadas, tales como la protección. Por ejemplo, las técnicas de virtualización permiten ver una única infraestructura física de diferentes puntos de vista. Esto simplifica mucho la tarea de los desarrolladores. Ahora no tienen que pensar en que switches tienen que instalar las reglas de *forwarding*, sino que ven toda la red como un gran switch.

2.5.7. Lenguajes de programación

Los diferentes lenguajes de programación se han estado proliferando durante décadas. Han evolucionado desde lenguajes de máquina de bajo nivel a lenguajes de alto nivel como java[18] y Python.

En SDN los lenguajes de programación de alto nivel pueden ser utilizados para:

1. Crear altos niveles de abstracción, simplificar la tarea de los dispositivos de red.
2. Habilitar entornos más productivos para acelerar el desarrollo y la innovación.
3. Promover la modulación del software y reutilizar el código en el plano de control de la red.
4. Fomentar el desarrollo de la virtualización de la red.

Los lenguajes de programación de alto nivel son herramientas muy poderosas. Se utilizan para implementar y proporcionar funciones como estructurar toda la red, alcalizaciones distribuidas, composición modular y virtualización.

18 *Java* es un lenguaje de programación de propósito general, concurrente y orientado a objetos. Fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible.

2.5.8. Aplicaciones de red

Son vistas como el cerebro de la red. Implementan la lógica de control que se traduce en comandos que se instalan en el plano de datos; controlando el comportamiento de los dispositivos de reenvío. Debe elegir la ruta a utilizar entre dos puntos y dar instrucciones al controlador para instalar las respectivas reglas.

Las SDN se pueden desplegar en una gran variedad de entornos, lo cual ha generado una amplia gama de aplicaciones de red. Las aplicaciones de red se pueden agrupar en cinco categorías: ingeniería de tráfico, movilidad e inalámbricas, medición y monitoreo, seguridad y confiabilidad y datos centrales de red. [8]

Siguiendo el artículo de *Intune Networks* [6]. Hay varias formas de aplicar SDN, tienen diferentes requisitos de red, y se dividen a grandes rasgos en dos grupos:

- SDN integrado (*embedded*): Esta es una forma que comparte la idea llamada purista ya que no propone un cambio total, sino implementar OpenFlow en los elementos de red. SDN integrado pretende resumir el control centralizándolo, para eso utiliza un protocolo como OpenFlow, para que controle los elementos de red. Así estos dispositivos, están controlados por las solicitudes del sistema central.

  Una de las características de este sistema, es que puede bajar considerablemente el precio de los switches debido a que el control en los mismos estará abstraído a la capa de control central. Aun así, tienen que soportar alguna de sus funciones originales.
- SDN superpuesto (*Overlaid*): Este enfoque implementa SDN dentro de máquinas virtuales en todo el área que lo utilice. Crea switches virtuales que se ejecutan en servidores informáticos finales; estos tienen los mismos requisitos de conectividad y privacidad que los físicos, y bajo el control de SDN. Crea un servicio unificado mediante la creación de túneles para llegar al destino. Sin embargo, para que la red sea transparente, debe

proporcionar la conectividad a cada dispositivo final sin congestión. La principal ventaja de SDN superpuesto es que no necesita una mejora en todos sus elementos de red.

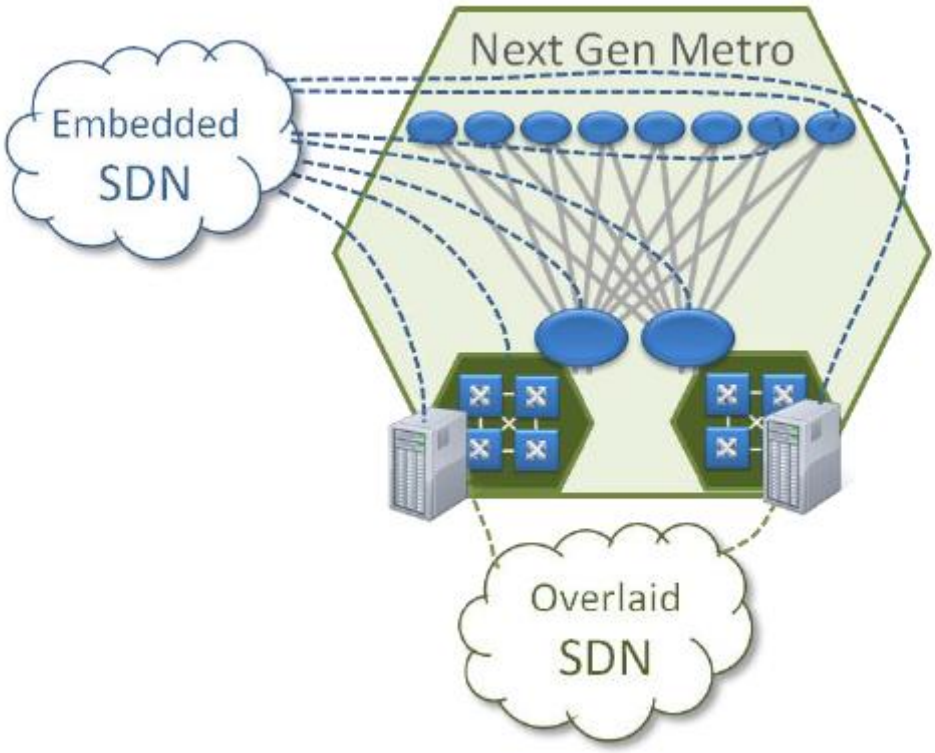

Figura 10: SDN integrado y SDN superpuesto.

Uno de los objetivos es estandarizar un protocolo único a todos los dominios ya que los protocolos existentes tienen el mismo objetivo. Esto simplifica las operaciones, disminuye gastos y elimina la opción de que un único fabricante sea el que pueda proveer determinados servicios, lo que ocasiona unos altos costes.

Con cualquiera de las dos formas de entender SDN el objetivo será unificar la red. Lo cual origina problemas entre la capa de control y la capa de datos, por eso se quiere separar el control de los dispositivos de red y dar esa función al controlador. Al querer que todo esté en la misma malla de conectividad al igual

que una LAN[19], en SDN superpuesto solo habría que extenderlo en la red y ya estaría estandarizado. Para SDN integrado tampoco representa un problema, ya que solo tendría que manejar un controlador centralizado en toda la red en lugar de varios dispositivos de red.

Esto se puede realizar gracias a una nueva tecnología de distribución híbrida llamada redes de sub-longitud de onda, como se ve en la figura 9. Se ha creado precisamente para poder tener un único switch distribuido, con puertos que puedan enviar paquetes a otros puertos aunque esté a una distancia muy alejada. [6]

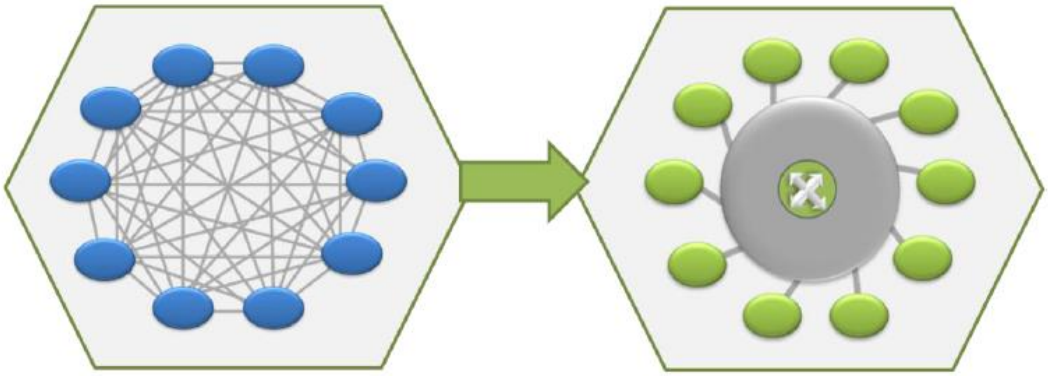

Figura 11: Conmutación distribuida.

2.6. Elementos para la selección del controlador SDN

Hay diferentes criterios para poder elegir entre un controlador y otro. La figura 12, muestra cuales son los criterios a seguir para decidir el controlador adecuado.

[19] Una red de área local, o LAN por las siglas en inglés de *local Area Network*, es una red de computadoras que abarca un área reducida a una casa, un departamento o un edificio.

Figura 12: Criterios hasta la elección del controlador adecuado.

A continuación se presentan cada una de estas pautas para la elección del controlador:

- Soporte OpenFlow: Es muy importante para el administrador de red a la hora de elegir el controlador, prestar atención a las versiones de OpenFlow que soporta y si es posible adaptarla a una nueva versión.
- Virtualización de red: la virtualización no es nada nuevo, ya existía por ejemplo la VLAN o VRF (*Virtual Routing Forwarding*). Una de las muchas ventajas del desacoplamiento de las redes virtuales y redes físicas, es que permite a las organizaciones de TI realizar cambios en la red física, como escalar la capacidad, sin afectar los flujos existentes. Otra de las muchas ventajas de la virtualización de red es que permite un aislamiento entre cada dispositivo de red, lo que es muy importante en temas de seguridad. También disminuye el número de servidores físicos, esto trae como consecuencia una reducción directa de los costos de mantenimiento de hardware.
- Funcionalidad de la red: con el fin de responder a la industria y multitud de regulaciones gubernamentales sobre la seguridad de datos, las organizaciones de TI a menudo necesitan mantener los datos generados por un conjunto de usuarios aislados de otros usuarios. Para conseguir eso, el controlador SDN debe habilitar las redes virtuales para que estén completamente aisladas unas de otras. Otra de las relaciones con el enrutamiento del tráfico, es la capacidad para descubrir múltiples caminos desde el origen a su destino para poder dividir el tráfico a través de varios enlaces.
- Escalabilidad: SDN permite a los desarrolladores de red agregar funcionalidades a la misma cuando sea necesario; esto es gracias al controlador SDN. Por ejemplo, un aspecto muy importante es saber

cuántos switches puede soportar el controlador. Otro factor que puede limitar la escalabilidad es la proliferación de entradas en la tabla de flujo, para ello se requiere una entrada salto por salto para cada flujo. También es necesario que el controlador pueda responder al impacto de sobrecarga de difusión de la red.

Otro aspecto relacionado con la escalabilidad, es que el controlador permita el movimiento de máquinas virtuales y el almacenamiento virtual entre sitios.

- Funcionamiento: la función principal del controlador es añadir flujos a la tabla. Los indicadores importantes son el tiempo de establecimiento del flujo y la cantidad de flujos por segundo que puede configurar. Esto tiene gran importancia cuando el controlador tiene que configurar más flujos de los que puede soportar.

  Los flujos se pueden configurar de dos maneras: de manera proactiva o reactiva. La manera proactiva tiene identificado que va a hacer con el flujo antes de que este llegue al switch OpenFlow. Con esto, el flujo se tarda en procesar un tiempo insignificante y casi no existe límite de flujos por segundo de este tipo, que el controlador puede procesar.

  La configuración reactiva se produce cuando el switch OpenFlow recibe un paquete y no coincide con ninguna entrada de flujo. El switch se lo envía al controlador para saber qué hacer con él, una vez que sabe dónde enviarlo, el paquete vuelve al switch con la información y esta se almacena en la cache por un tiempo limitado. El tiempo asociado a la configuración de un flujo reactivo es la suma del tiempo que se necesita para enviar el paquete del switch al controlador, el tiempo que tarda en procesar el paquete el controlador, el que se tarda en reenviar el paquete con la información necesaria de vuelta al switch y el tiempo en el que se inserta el flujo en la tabla. Los factores que afectan al tiempo de establecimiento del flujo incluyen la capacidad de procesamiento del switch y el procesamiento y rendimiento de entrada / salida del controlador. Por ejemplo un controlador cuyo software este escrito en C será más rápido que otro que este escrito en Java.

- Programación de red: En las redes tradicionales, la configuración de red se realiza dispositivo a dispositivo. Este enfoque es lento y propenso a errores, esto corresponde a que las redes tradicionales son muy estáticas y esto perjudica en el potencial de rendimiento de la red.

  Una de las características fundamentales de SDN, es que hay interfaces de programación en el controlador. Ejemplos de programación que se deben buscar en un controlador SDN son la capacidad de redirigir el tráfico (por razones de seguridad). Se puede elegir que el tráfico entrante pase a través de un cortafuegos o *firewall*[20]. Estos crean filtros que son considerados ACLs[21] dinámicas e inteligentes como combinaciones complejas de múltiples campos de cabecera de paquetes. Un controlador SDN también puede aportar la programabilidad, proporcionando plantillas que permitan la creación de secuencias de comandos CLI[22], con las que es posible la programación dinámica de la red.

20 Un cortafuegos (*firewall*) es una parte de un sistema o una red que está diseñada para bloquear el acceso no autorizado, permitiendo al mismo tiempo comunicaciones autorizadas

21 Una lista de control de acceso o ACL (del inglés, *access control list*) es un concepto de seguridad informática usado para fomentar la separación de privilegios. Es una forma de determinar los permisos de acceso apropiados a un determinado objeto, dependiendo de ciertos aspectos del proceso que hace el pedido.

22 La interfaz de línea de comandos, traducción del inglés *command-line interface* o CLI .

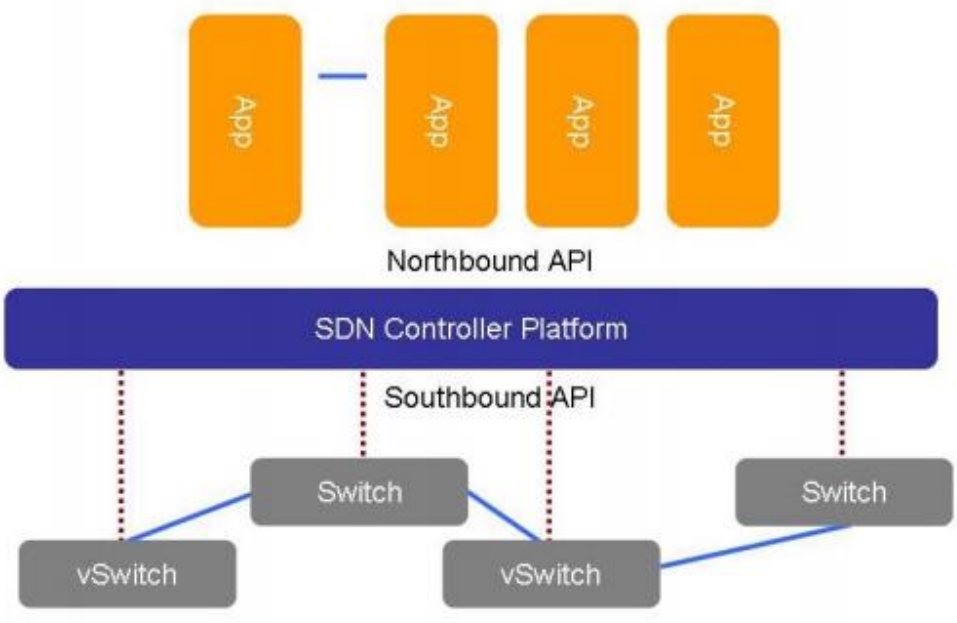

Figura 13: Arquitectura SDN en capas.

Otra forma de permitir programabilidad es con una API orientada al norte como se ve en la figura 13. Hace que la información se centralice en el controlador a disposición de un conjunto de aplicaciones SDN, que son capaces de reenviar paquetes a través de la ruta menos costosa o cambiar la configuración de *QoS*[23]. Estas aplicaciones ofrecen servicios como *firewall* y balanceadores de carga.

- Fiabilidad: Una parte muy importante en la implantación de SDN es que el controlador realiza una validación del diseño, lo que elimina los errores manuales. Una técnica que un controlador SDN puede utilizar para aumentar la fiabilidad de la red, es la capacidad para descubrir múltiples caminos entre el origen y el destino. Esto es útil en el caso de que se corte un enlace fallido, el controlador es capaz de redirigir el tráfico rápidamente.

23 *QoS* o Calidad de Servicio (*Quality of Service*) es el rendimiento promedio de una red de telefonía o de computadoras, particularmente el rendimiento visto por los usuarios de la red.

Con respecto a las conexiones externas, es importante que el controlador soporte tecnologías como el VRRP[24], que tienen como objetivo aumentar la fiabilidad de la red. Es imprescindible que el controlador SDN permita agrupaciones *cluster*[25]. Si hay varios controladores en *cluster* sincronizados entre sí proporcionará mayor fiabilidad a la red.

- Seguridad de la red: Para proporcionar seguridad a la red, un controlador SDN debe ser capaz de soportar la autenticación y autorización del administrador de la red. El administrador de red también debe ser capaz de bloquear el acceso a determinadas aplicaciones. Esto se puede conseguir poniendo en práctica las ACL inteligentes y dinámicas. Un aspecto muy importante es que cada cliente que comparte infraestructura con otros, tenga un aislamiento total de los otros.
- Monitorización centralizada y visualización: Una de las principales ventajas de SDN, es que permite a las organizaciones de TI tener una visibilidad de flujo extremo a extremo.

  Un controlador SDN tiene que ser capaz de utilizar los datos ofrecidos por el protocolo OpenFlow para identificar los problemas en la red y, automáticamente, cambiar la ruta que toma un flujo determinado. El controlador también tiene que permitir a las organizaciones de TI controlar unas clases de tráfico y otras no, por ejemplo, puede optar por no controlar su tráfico de respuesta.

  El controlador SDN debe poder presentar a una organización de TI una visualización de los enlaces físicos de la red. Además de presentar las

24 *Virtual Router Redundancy Protocol* (*VRRP*) es un protocolo de redundancia no propietario definido en el RFC 3768, diseñado para aumentar la disponibilidad de la puerta de enlace por defecto dando servicio a máquinas en la misma subred.

25 El término clúster (del inglés *cluster*, "grupo" o "raíz") se aplica a los conjuntos o conglomerados de computadoras unidos entre sí normalmente por una red de alta velocidad y que se comportan como si fuesen una única computadora.

múltiples redes virtuales que se ejecutan en la red, también tiene que ser capaz de ver los flujos tanto de la perspectiva física como de la virtual y descubrir detalles de ellos.

También, debería ser posible monitorizar el controlador SDN utilizando protocolos y técnicas estándares de gestión tales como SNMP[26]. Adicionalmente el controlador SDN debe ofrecer soporte para una amplia gama de MIBs[27] estándares y MIB privadas para poder controlar los elementos de la red virtual. Lo ideal sería que presentara la información de la red en una API REST.

- Fabricantes de controladores SDN: debido al creciente interés en las SDN, numerosos vendedores han entrado en el mercado y otros más han anunciado su intención de hacerlo. Debido a la volatilidad del mercado SDN en general, y del mercado del controlador SDN en particular, los fabricantes están evaluando que no solo se deben centrar en aspectos técnicos del controlador, sino también en: la competencia del vendedor, la resistencia global del proveedor así como la profundidad de la organización de su ingeniería.
- Otro aspecto fundamental, es la capacidad de recuperación del vendedor en el mercado SDN. Una empresa de controladores SDN que sale al mercado, si hace una mala elección de proveedor, genera un daño a la empresa durante por lo menos 1 o 2 años. Sin embargo, si sus proveedores son grandes se pueden recuperar de alguna caída en el mercado y mantener la empresa. [9]

2.7. Controladores

26 El Protocolo Simple de Administración de Red o *SNMP* (del inglés *Simple Network Management Protocol*) es un protocolo de la capa de aplicación que facilita el intercambio de información de administración entre dispositivos de red.

27 La Base de Información Gestionada (*Management Information Base* o *MIB*) es un tipo de base de datos que contiene información jerárquica, estructurada en forma de árbol, de todos los dispositivos gestionados en una red de comunicaciones.

Las SDN proponen una forma de gestionar las redes en la cual el control se desprende del hardware y como se explica anteriormente se da el mismo a una aplicación de software llamada controlador. Otro de los propósitos de SDN es flexibilizar la compatibilidad entre dispositivos de diferentes fabricantes, por eso mismo hay numerosos controladores de código abierto.

Un controlador SDN / OpenFlow para redes es como un sistema operativo para ordenadores. Pero el estado actual del mercado de controlador SDN / OpenFlow puede ser comparado con los sistemas operativos que existían hace 40-50 años. Los sistemas operativos eran software de control que proporcionaba bibliotecas de código de soporte. Esto se hacía para ayudar a los programas con operaciones de tiempo de ejecución.

En la actualidad existen más de 30 controladores diferentes creados por universidades, grupos de investigación o vendedores. Cada uno de ellos posee unas características propias.

A continuación se presentan algunos de los controladores y sus principales características.

2.7.1. NOX

NOX es una parte del ecosistema de SDN. En concreto, se trata de una plataforma para la creación de aplicaciones de control de red. De hecho, mientras que lo que ahora llamamos SDN nació de una serie de proyectos académicos, la primera tecnología SDN para obtener el reconocimiento verdadero del nombre fue OpenFlow y NOX.

Se desarrolló inicialmente en *Nicira*[28], desarrollado al mismo tiempo que OpenFlow, NOX fue el primer controlador OpenFlow. Nicira donó NOX a la comunidad de investigadores en 2008, y desde entonces ha sido la base de

28 *Nicira* era una empresa enfocada en la creación de redes definidas por software (SDN) y virtualización de la red. Fue fundada en 2007 por Martín Casado, Nick McKeown y de Scott Shenker. *Nicira* creo sus propias versiones de propiedad de OpenFlow, *open vSwitch* y proyectos de redes *OpenStack*.

múltiples y diversos proyectos de investigación a principios de la investigación SDN.

Para un desarrollador, NOX:

- Proporciona una API en C++ OpenFlow 1.0.
- Proporciona rapidez de entrada/salida asíncrona.
- Está dirigido a las distribuciones recientes de Linux (especialmente Ubuntu[29] 11.10 y 12.04, pero en Debian[30] también funciona, y en RHEL[31] es posible)
- Incluye ejemplos compuestos para: El descubrimiento de la topología, *learning switch* y *network-wide switch*.

Todo esto va a dar a los desarrolladores e investigadores una manera de escribir código. Se podrá controlar mediante la programación de switches (tanto de hardware como virtuales) en sus redes.

El funcionamiento de NOX, como se observa en la figura 19, es a través de eventos. Esto quiere decir que cualquier proceso que llega al controlador, como la conexión de un dispositivo o su desconexión, genera un evento que permite controlar este suceso.

29 Ubuntu es un sistema operativo basado en GNU/Linux y que se distribuye como software libre.

30 Debian o Proyecto Debian (en inglés: *Debian Project*) es una comunidad conformada por desarrolladores y usuarios, que mantiene un sistema operativo GNU basado en software libre.

31 *Red Hat Enterprise Linux* también conocido por sus siglas RHEL es una distribución comercial de Linux desarrollada por Red Hat.

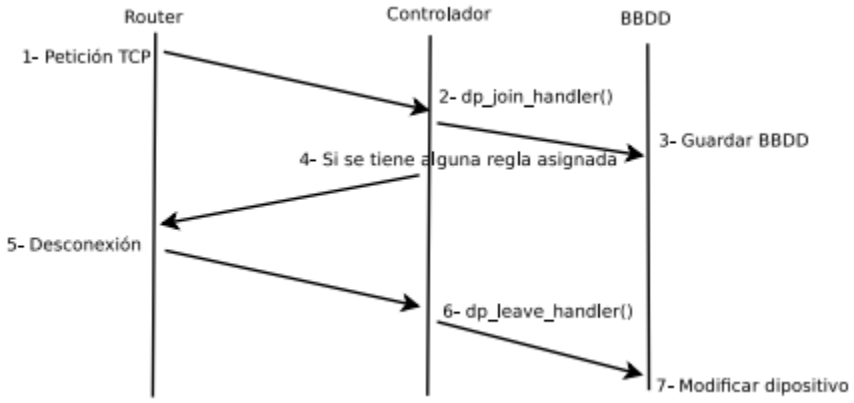

Figura 14: Diagrama de eventos NOX

Estos son los pasos que se ven en la figura 19:

1. Petición TCP. La configuración OpenFlow del router inicia una conexión TCP con el controlador, en caso de no estar lo reintenta hasta que consigue conectarse
2. Evento *dp_join_handle*. Es un evento que lanza NOX cuando se conecta cualquier dispositivo.
3. Se guardan los datos en la base de datos.
4. Se envían las reglas asignadas a ese puerto o IP (si tiene asignada alguna acción).
5. Se produce la desconexión del router.
6. El controlador lanza el evento *dp_leave_handler*, el cual se encarga de enviar el identificador a la base de datos. Indica en la base de datos que ese dispositivo está disponible. [10]

2.7.2. POX

POX es el hermano menor de NOX. Básicamente, es una plataforma para el desarrollo rápido y creación de prototipos de software de control de red utilizando Python. Es uno de los marcos crecientes (incluyendo NOX, Floodlight, etc.) para facilitar la creación de un controlador OpenFlow.

POX tiene más características y posibilidades. Además de ser un marco para interactuar con switches OpenFlow, se está usando como base para algunos trabajos en curso como ayudar a construir la disciplina emergente de SDN,

explorar y distribuir el prototipo, depuración SDN, virtualización de redes, diseño del controlador, y modelos de programación. El objetivo final es desarrollar un controlador SDN moderno.

POX está aún bajo desarrollo. La base principal es la investigación, y muchos proyectos de tienen una vida bastante corta, por eso POX se centra en mantener una API estable.

Algunas de las funciones de POX son las siguientes:

- Interfaz OpenFlow "*Pythonic*".
- Componentes reutilizables para la selección de rutas, el descubrimiento de topología, etc.
- Funciona en cualquier lugar. Se puede combinar con instalar *PyPy*[32] para una fácil implementación.
- En concreto se dirige a Linux, Mac OS y Windows.
- Soporta los mismos GUI[33] y herramientas de visualización que NOX.
- Se desempeña bien en comparación con las aplicaciones de NOX escritas en Python (especialmente cuando se ejecuta en *PyPy*)

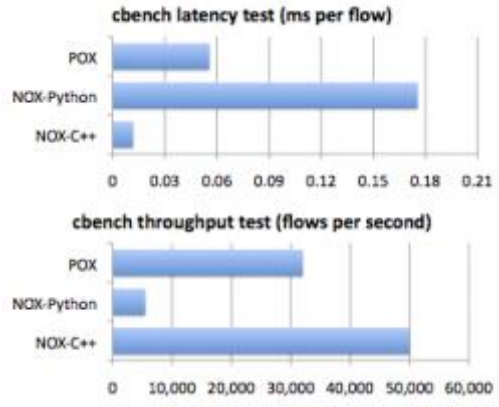

Figura 15: Test comparativo de POX con NOX en ms por flujo y flujos por segundo.

32 *PyPy* es un intérprete y compilador para el lenguaje Python, que se enfoca en la velocidad y eficiencia, y es 100% compatible con el intérprete original *CPython*.

33 La interfaz gráfica de usuario, conocida también como *GUI* (del inglés *Graphical User Interface*) es un programa que actúa de interfaz de usuario.

Aunque POX todavía es prematuro, ya ha tenido algún uso en la investigación y la educación. [11]

2.7.3. BEACON

C y C ++ fueron los lenguajes de programación principales de alguno de los controladores de OpenFlow antes de Beacon, por ejemplo NOX está escrito en C++. Estos lenguajes no solo se pueden usar para producir aplicaciones de muy alto rendimiento, sino que también vienen con una significativa carga de desarrollo. Los problemas más comunes incluyen:

- Es pesado (> 10 minutos) en tiempo de compilación.
- Los errores de compilación enturbian el propósito real.
- Hay errores de programación de gestión de memoria manuales que conducen a la segmentación fallos.
- Pérdidas de memoria

Los enfoques han tratado de minimizar algunos de estos problemas. Hay componentes para disminuir el tiempo de compilación y el uso estricto de técnicas (como punteros inteligentes) sirve para mejorar los errores de memoria. La elección de C / C ++ para algunos entornos, es la decisión correcta, pero otros lenguajes más manejables podrían ser utilizados para crear un controlador OpenFlow de alto rendimiento. Esto está destinado a funcionar en el hardware de los productos básicos, donde la CPU y RAM incrementan fácilmente su precio.

Beacon es un controlador OpenFlow de código abierto basado en Java creado en 2010. Ha sido ampliamente utilizado para la enseñanza, la investigación y como la base de Floodlight.

Algunas de sus características principales son:

- Estabilidad. Beacon ha estado en desarrollo desde principios de 2010, ha sido utilizado en varios proyectos de investigación, cursos de redes y pruebas de implementaciones. Actualmente Beacon acciona un 100-

vswitch, centros de datos experimentales de 20-switches físicos y ha estado durante meses sin tiempo de inactividad.

- Multiplataforma. Beacon está escrito en Java y se ejecuta en muchas plataformas, desde los servidores Linux *multi-core* de gama alta hasta los teléfonos Android.
- Es de código abierto. Beacon está licenciado bajo una combinación de la GPL[34] v2 *license* y Stanford University FOSS (*Free and Open Source Software*) *License Exception* v1.0.
- Dinámico. Los paquetes de código en Beacon se pueden iniciar / parar / actualizar / instalar en tiempo de ejecución, sin interrumpir otros paquetes no dependientes (es decir, reemplazar la aplicación *Learning Switch* en marcha sin necesidad de desconectar los switches).
- Desarrollo rápido. Es fácil de poner en marcha. Java y Eclipse[35] simplifican el desarrollo y la depuración de sus aplicaciones.
- Rápido. Beacon es multiproceso.
- Interfaz de usuario Web. Incorpora opcionalmente el servidor web de la empresa Jetty[36], además un marco de interfaz de usuario personalizada extensible.
- Beacon se basa en los marcos de Java maduros como Spring[37] y Equinox[38] (OSGi[39]).

---

34 La Licencia Pública General de GNU o más conocida por su nombre en inglés GNU *General Public License* (o simplemente sus siglas del inglés *GNU GPL*) es la licencia más ampliamente usada en el mundo del software y garantiza a los usuarios finales (personas, organizaciones, compañías) la libertad de usar, estudiar, compartir (copiar) y modificar el software.

35 Eclipse es un programa informático compuesto por un conjunto de herramientas de programación de código abierto multiplataforma.

36 Jetty es un servidor *HTTP* 100% basado en *Java* y un contenedor de *Servlets* escrito en Java.

37 *Spring* es un *framework* para el desarrollo de aplicaciones y contenedor de inversión de control, de código abierto para la plataforma *Java*.

38 *Equinox* ofrece soluciones de código abierto para librerías de todo el mundo.

La mayoría de los controladores de OpenFlow tienen la capacidad de seleccionar las aplicaciones para compilar, poseen modularidad de tiempo y de las aplicaciones para lanzar cuando se inicia el controlador.

Beacon es capaz de no sólo iniciar y detener aplicaciones mientras se está ejecutando, sino también agregarlas y retirarlas sin necesidad de cerrar el proceso de Beacon. Esto permite nuevas formas para que los desarrolladores puedan interactuar con las instancias Beacon desplegados.

Los desarrolladores determinan cómo de modular es su aplicación. Por ejemplo, un paquete puede contener múltiples aplicaciones o una sola y se puede propagar a través de múltiples paquetes. Estas decisiones suelen hacerse sobre el grado de modularidad que tiene una aplicación. Por ejemplo, si una parte de la aplicación podría ser sustituida en el arranque o en tiempo de ejecución. Tal como el motor de enrutamiento siendo utilizado por aplicación de enrutamiento de Beacon.

El rendimiento de un controlador de OpenFlow se mide típicamente como el número de paquetes que un controlador puede procesar y responder por segundo. También como el tiempo promedio que el controlador toma al procesar cada evento.

Beacon probó nuevas áreas del espacio de diseño del controlador OpenFlow; con un enfoque manejable para el desarrollado, de alto rendimiento, y tiene la capacidad para iniciar y detener aplicaciones (nuevas o existentes) en el tiempo de ejecución. Beacon sorprendentemente mostró alto rendimiento. Maneja 12,8 millones de paquetes en mensajes por segundo con 12 núcleos, mientras que además se construye utilizando Java. [12]

---

39 *OSGI* son las siglas de *Open Services Gateway Initiative*. Su objetivo es definir las especificaciones abiertas de software que permita diseñar plataformas compatibles que puedan proporcionar múltiples servicios

2.7.4. FLOODLIGHT

Floodlight es un controlador SDN abierto. Está probado y apoyado por la comunidad de desarrolladores de controladores SDN más grande del mundo. Cabe destacar que fue diseñado para ser fácil de usar y configurar, con mínimas dependencias y *developer* también sencillo. Al mismo tiempo ofrece un sistema modular que hace que sea fácil de desarrollar y mejorar. Floodlight admite una amplia gama de switches virtuales *hypervisor-based* como Open *vSwitch*. O como el creciente ecosistema de switches OpenFlow físicos con alta compatibilidad con OpenFlow mixto y redes no OpenFlow.

Y es que los desarrolladores OpenStack[40] pueden ahora conectarse a Floodlight en redes programables en la nube *multi-tenant*. También pueden vincularse con entornos virtualizados con el *Quantum Plugin* que fue aportado a finales de 2012.

Las empresas y organizaciones que se han descargado las API de Floodlight incluyen: Arista, Brocade, Citrix, Dell, Extreme Networks, Fujitsu, Google, HP, IBM, Intel, Juniper Networks and Microsoft, entre otros. [13]

2.8.4.1. ¿Por qué utilizar Floodlight?

- OpenFlow. Funciona con switches físicos y virtuales que entienden el protocolo OpenFlow.
- La licencia Apache[41] que Permite usar el Floodlight para casi cualquier propósito.
- Es una comunidad abierta. Floodlight se lleva a cabo por una comunidad abierta de desarrolladores. Permite contribuciones al código de participantes activos y comparte abiertamente información sobre el estado del proyecto, plan de trabajo, *bugs*, etc.

---

40 *OpenStack* es un proyecto de computación en la nube para proporcionar una infraestructura como servicio.

41 La licencia *Apache (Apache License o Apache Software License* para versiones anteriores a 2.0) es una licencia de software libre creada por la *Apache Software Foundation (*ASF*)*

- Probado y apoyado. Floodlight es el núcleo de un controlador comercial producto de *Big Switch Networks* y está probado y mejorado por una comunidad de desarrolladores profesionales.

El controlador Floodlight SDN es un controlador OpenFlow de clase empresarial, con licencia de Apache y basado en Java. Esta apoyado por una comunidad de desarrolladores que incluyen una serie de ingenieros *de Big Switch Networks.*

OpenFlow es un estándar abierto gestionado por la Open Networking Foundation. Se especifica un protocolo a través de un switch, un controlador remoto puede modificar el comportamiento de los dispositivos de red a través de un "conjunto de instrucciones de *forwarding* " bien definidas. Floodlight está diseñado para trabajar con el creciente número de switches, routers, virtual switches, y los puntos de acceso compatibles con el estándar OpenFlow.

2.8.4.2. Características más importantes

- Ofrece un sistema de módulo de carga que hacen que sea fácil de expandir y mejorar.
- Fácil de configurar con dependencias mínimas.
- Soporta una amplia gama de switches OpenFlow virtualización y físico. Puede manejar tanto redes OpenFlow como no OpenFlow conjuntamente. Además permite gestionar múltiples "islas" de switches con hardware OpenFlow.
- Está diseñado para ser de alto rendimiento. Es el núcleo de un producto comercial de *Big Switch Networks*. [12]

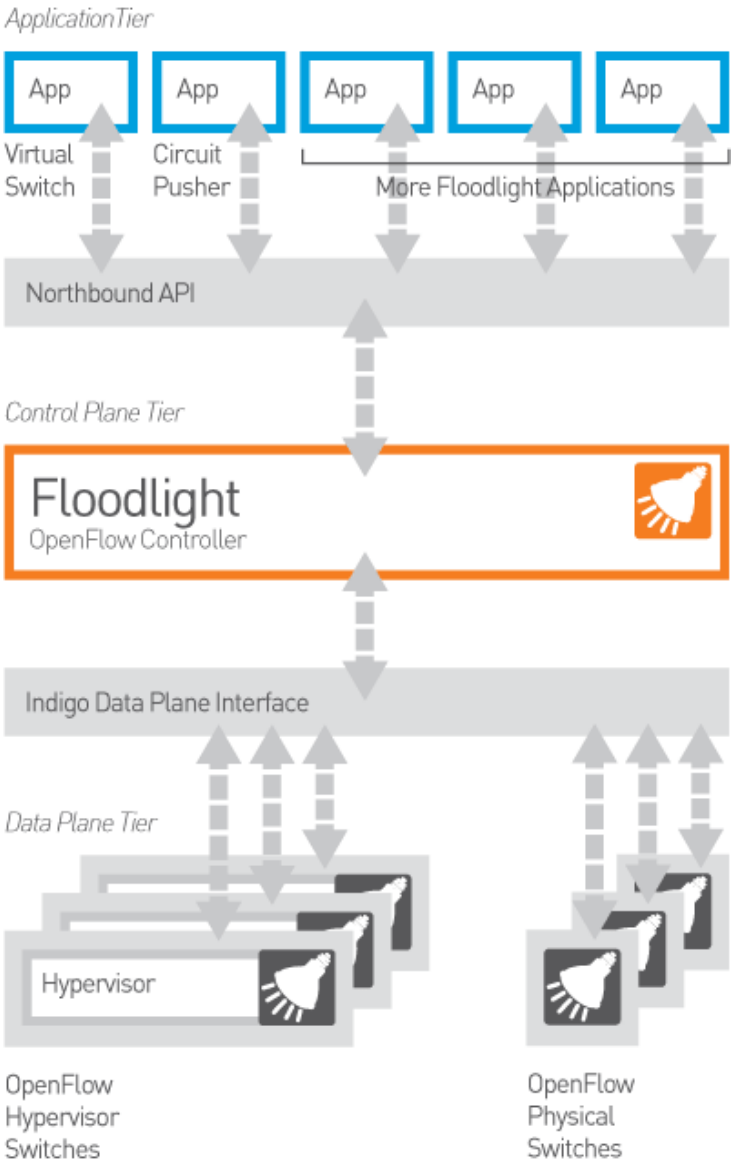

Figura 16: Funcionamiento del controlador Floodlight.

2.8.4.3. Arquitectura de Floodlight

Floodlight no es sólo un controlador OpenFlow, sino que también es una colección de aplicaciones construidas además del Controlador Floodlight.

El controlador Floodlight comprende un conjunto de funcionalidades comunes para controlar e investigar una red OpenFlow. Además las aplicaciones comprenden diferentes características para resolver distintas necesidades de los usuarios en la red.

La figura 22 muestra la relación entre el controlador del Floodlight, las aplicaciones (construidas como módulos Java compiladas con Floodlight) y las aplicaciones construidas sobre la API REST Floodlight.

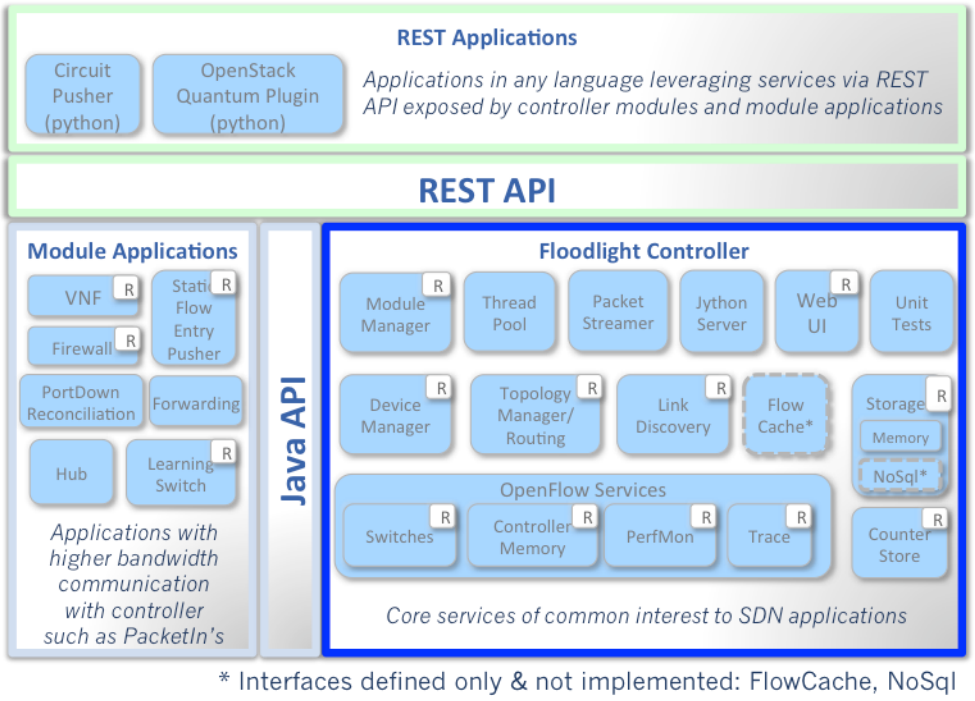

Figura 17: Arquitectura Floodlight.

2.8.4.4. Topologías que soporta

Floodlight actualmente cuenta con dos aplicaciones de reenvío de paquetes reactivos. Ambas, tienen diferentes comportamientos y trabajan con topologías como se describe a continuación:

1. Con *forwarding*. Esta opción está habilitada por defecto nada más empezar. *Forwarding* permite el reenvío de paquetes de extremo a extremo entre dos dispositivos en las siguientes topologías de red:
    - Dentro de una isla OpenFlow[42]. Cuando cualquier dispositivo A envía un paquete hacia el dispositivo B en la misma isla

42 Los términos "islas" y "*cluster*" se utilizan indistintamente. Una isla / *cluster* OpenFlow es un conjunto de switches OpenFlow con los dispositivos conectados a cualquiera de ellos. Análogamente, una isla / *cluster* no OpenFlow es un conjunto de no OpenFlow switches con dispositivos conectados a cualquiera de ellos.

OpenFlow, *forwarding* calcula así el camino más corto entre A y B como muestran las figuras:

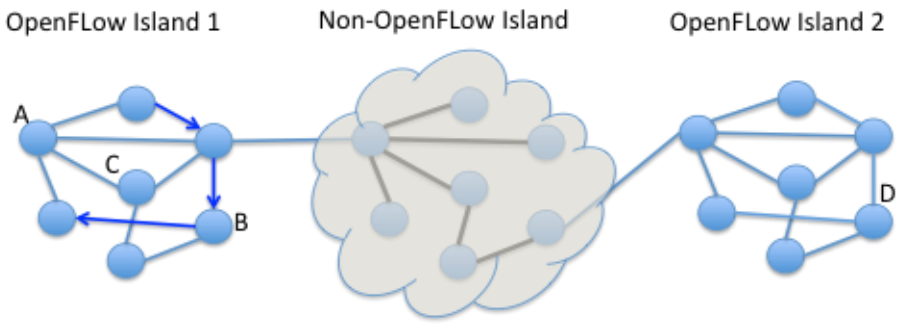

Figura 18: Ejemplo de topología de red que funciona con Floodlight.

- Islas OpenFlow con islas no OpenFlow entre ellas. Cada isla OpenFlow puede tener exactamente un nexo de unión a una isla no OpenFlow. Además, OpenFlow y las islas no OpenFlow juntas pueden no formar uniones, cada dispositivo tendría un punto de unión en cada isla OpenFlow. *Forwarding* calcula el camino más corto en cada isla OpenFlow y espera a que los paquetes sean enviados a las islas no OpenFlow.

Esto se puede comprobar más gráficamente con las siguientes figuras de ejemplo:

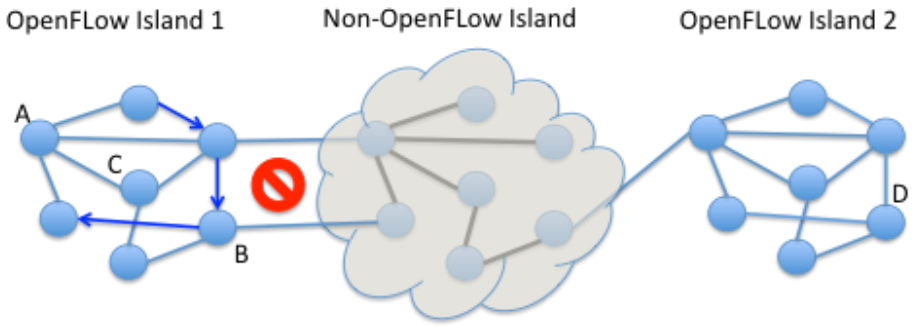

Figura 19: Ejemplo de topología de red que no funciona con Floodlight, a pesar de que cada isla OpenFlow está conectado a una isla no OpenFlow través de un solo enlace.

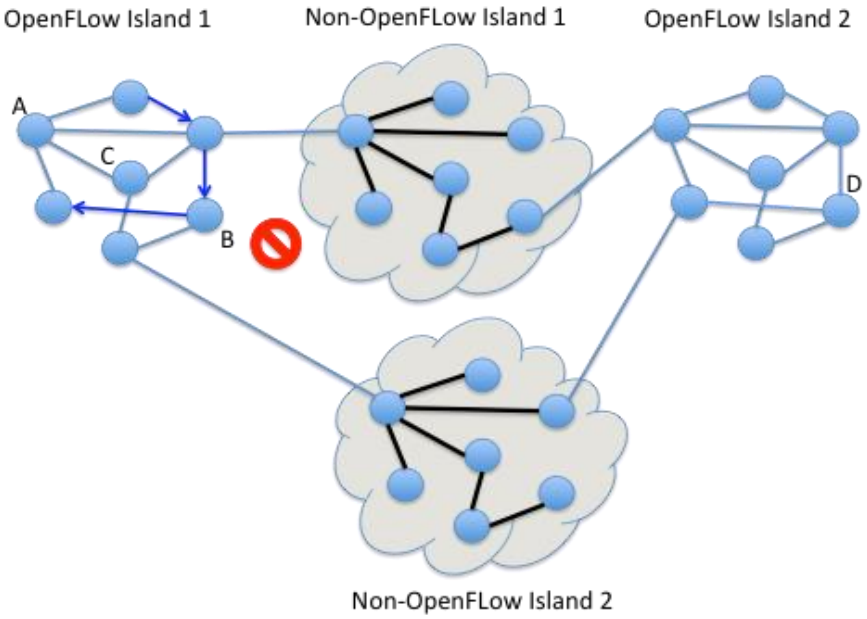

Figura 20: Ejemplo de topología de red que no funciona con Floodlight, tiene dos enlaces de la isla OpenFlow 1 que conectan con la isla no OpenFlow.

Los flujos instalados con tiempo *forwarding* de espera pueden expirar. Esto se debe a que no hay tráfico enviado a través de la ruta de acceso por un tiempo mayor al de timeout.

2. Con *Learning Switch*:
    - Está recomendado para su uso con cualquier número de islas OpenFlow, incluso con islas no OpenFlow entre ellas.
    - No puede funcionar si se conecta una isla con topología de anillo a otra isla. Tampoco si esta se conecta con otras islas con topología de anillo.
    - El rendimiento de reenvío es mucho menos eficiente que con los otros métodos.

Floodlight también proporciona una aplicación de *Static Flow Entry Pusher* y una aplicación *Circuit Pusher*, que permiten a los usuarios instalar por su cuenta rutas de reenvío en la red:

- *Static Flow Entry Pusher* permite instalar entradas de flujo switch por switch, creando así las rutas de reenvío basadas en la elección de puertos del switch por el usuario.

- *Circuit Pusher* se basa en *Static Flow Entry Pusher*, el *Device Manager* y servicios de enrutamiento basados en su API REST. Esto le sirve para construir circuitos con la ruta más corta, individuales dentro de una sola isla OpenFlow. [14]

2.7.5. RYU

Ryu es un software basado en componentes definido en un marco de red. Ofrece componentes de software con una API bien definida; esto hace que los desarrolladores interpreten fácilmente las aplicaciones de gestión y control de red. Ryu también es compatible con varios protocolos para la gestión de dispositivos de red, tales como OpenFlow y NETCONF.

En lo referente a OpenFlow, Ryu es totalmente compatible con 1.0, 1.2, 1.3, 1.4 y Nicira Extensions y está completamente escrito en Python. [15]

Los switches tienen una gran variedad de funciones:

- Conocer la dirección MAC de un host conectado a un puerto y guardarla en la tabla de direcciones MAC.
- Enviar un paquete al puerto de un host (si ese host es conocido).
- Si un paquete se dirige a una dirección desconocida, el switch inserta en todas las tablas el flujo del nuevo destino.

Los switches OpenFlow realizan las siguientes acciones tras la recepción de instrucciones de los controladores RYU:

- Reescriben la dirección de los paquetes recibidos o transfieren los paquetes desde el puerto especificado.
- Transfieren los paquetes al controlador (*Packet-in*).
- Envían los paquetes por el controlador desde el puerto especificado (*Packet-out*).

En primer lugar, es necesario utilizar la función Packet-in para obtener las direcciones MAC del host y la información del puerto al que está conectado. Cuando ya tiene todas las direcciones, el switch transfiere los paquetes que ha

recibido. A continuación el switch busca las direcciones para ver si las MAC de los paquetes pertenecen a los host que tenía conocidos. Con el resultado realiza uno de los siguientes procedimientos:

- Si el host ya estaba en la lista, ejecuta la opción *Packet-out* y envía los paquetes.
- Si el host es desconocido, utiliza *Packet-out* para inundar las listas.

En el siguiente ejemplo se explica gráficamente el procedimiento anterior:

1. Estado inicial: la tabla de flujo está vacía.

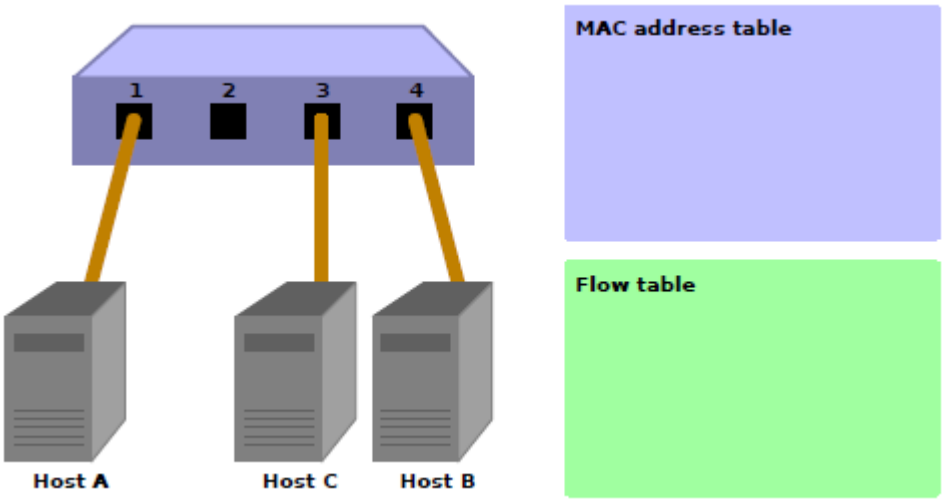

Figura 21: Switch OpenFlow ejemplo 1.

2. Cuando se envía un paquete del host A al host B, se envía un *Packet-in* y la dirección MAC del host A se guarda en la tabla de flujo. Como no se conoce la dirección del host B, se envía el paquete a los demás puertos, y este es recibido por el host B y el host C.

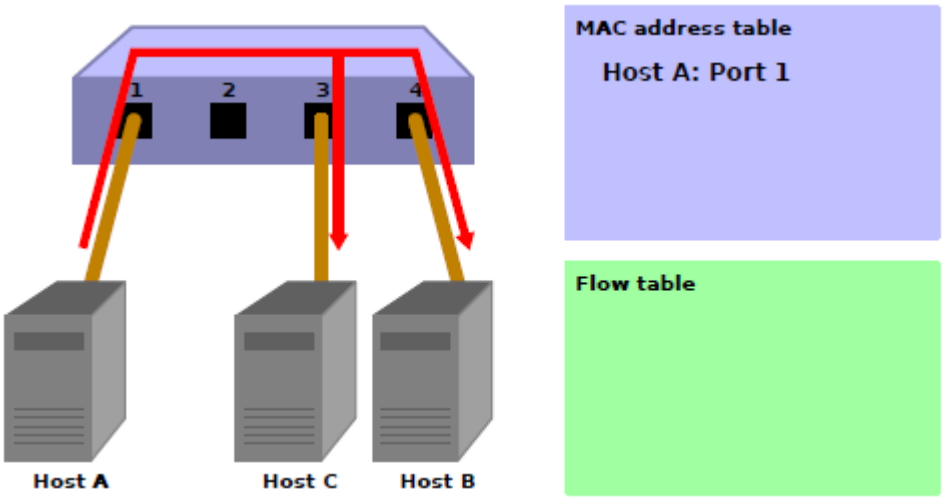

Figura 22: Switch OpenFlow ejemplo 2.

3. Cuando se devuelven los paquetes del host B al host A, se transfieren los paquetes al puerto 1 y se añade una entrada a la tabla de flujo. En esta ocasión, el host C no recibe los paquetes.

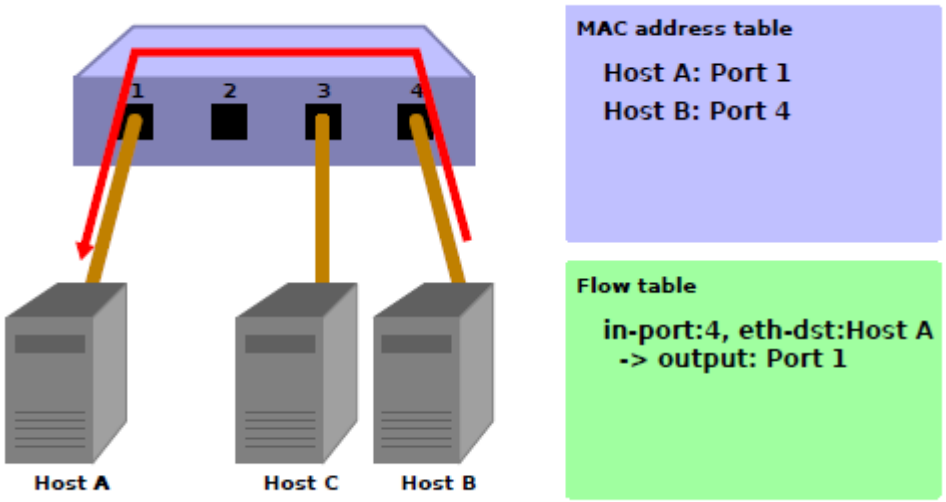

Figura 23: Switch OpenFlow ejemplo 3.

4. Cuando se vuelven a enviar otra vez los paquetes, del puerto 1 al puerto 4, se añade otra entrada de flujo y los paquetes se transfieren al puerto 4. [16]

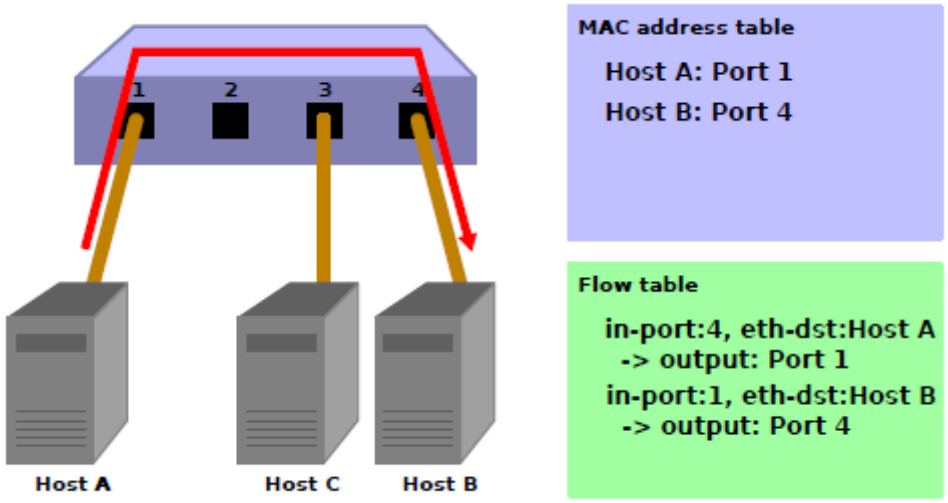

Figura 24: Switch OpenFlow ejemplo 4.

Capítulo 3. PROTOCOLO OPENFLOW

3.1. Introducción

OpenFlow surgió como parte de la investigación de la red de Stanford. Su idea inicial era la experimentación con protocolos que pudieran ser usados para la investigación. Comenzaron desde cero y fueron evolucionando con la intención de reemplazar las funcionalidades de las capas 2 y 3 de los switches y routers comerciales.

En 2011 se formó la fundación *Open Networking* (ONF) con el objetivo de comercializar y estandarizar el uso de OpenFlow. Esta organización ya tiene una estructura más comercial con un departamento de marketing encargado de promover el protocolo OpenFlow y SDN.

Los principios de OpenFlow son:

- Separación del plano de control y de datos
- Estandarizar un protocolo entre dispositivos de red y controlador
- Promover la programabilidad de la red desde una API centralizada.

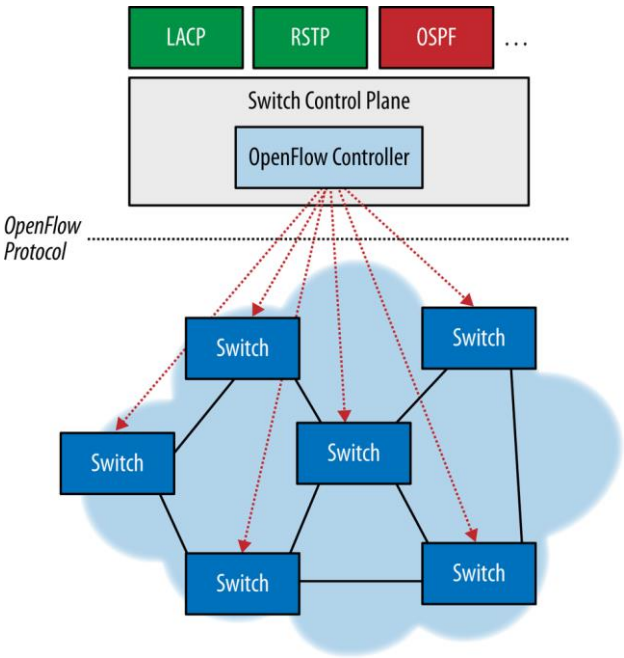

Figura 25: Modelo OpenFlow.

Los protocolos OpenFlow actualmente se dividen en dos partes:

- Protocolo de conexión.

  Las entradas de flujo ya no se almacenan en los dispositivos de red de forma permanente. El objetivo final es crear un estado de reenvío con la función de mantener unas reglas de envío de los paquetes.

  El concepto original era que el switch fuera capaz de comportarse como un dispositivo de servicio. En los sistemas basados en hardware esto dependía mucho del proveedor, sin embargo con la versión 1.3 de OpenFlow es posible que un elemento OpenFlow emule comportamientos de una plataforma OpenFlow integrada como MPLS.

- Configuración y extensión

  El protocolo se estructura en torno a mensajes XML[43], modelos de datos *Yang*[44] y el protocolo NETCONF[45] para la entrega y a partir de la versión 1.1 de *of-config* cada switch físico puede tener varios switch lógicos internos.[1]

43 XML, siglas en inglés de *eXtensible Markup Language* ('lenguaje de marcas extensible'), es un lenguaje de marcas desarrollado por el *World Wide Web Consortium* (W3C) utilizado para almacenar datos en forma legible.

44 *YANG* es un lenguaje de modelado de datos para *NETCONF*.

45 El Protocolo de configuración de red (*NETCONF*) es una gestión de la red de protocolo desarrollado y estandarizado por la *IETF*.

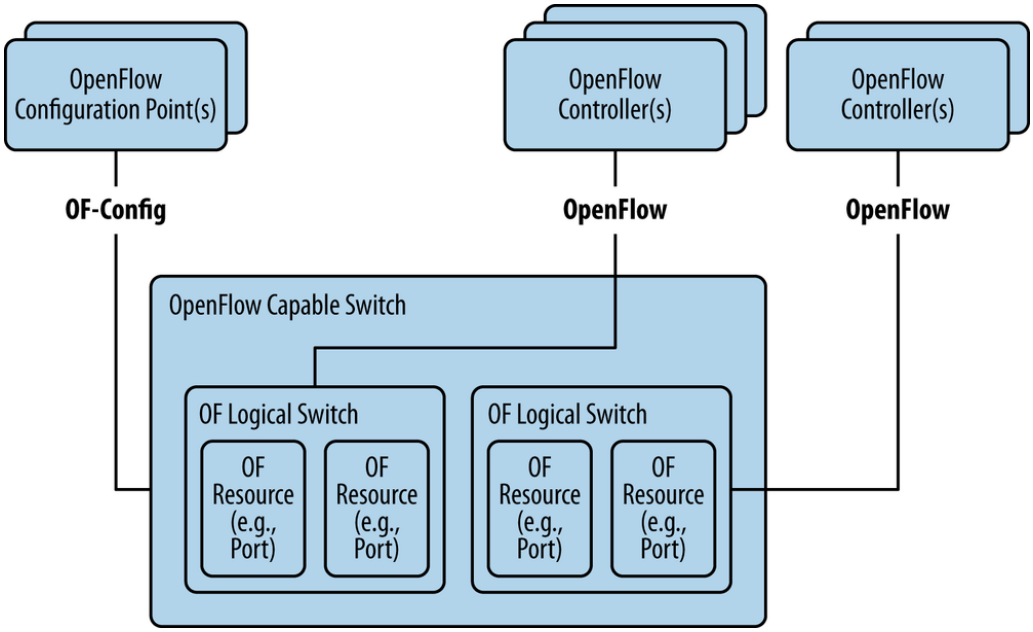

Figura 26: Switch físico y switch lógico.

3.2. Switch OpenFlow

Los switches tradicionales contienen tablas de flujo que trabajan para implementar firewalls NAT[46]. Aunque cada uno se rige por un fabricante, se han observado características comunes a todos ellos. Lo que pretende OpenFlow es crear un protocolo abierto para poder programar las diferentes tablas de flujo en los switches y routers.

El camino de datos de un switch OpenFlow consiste básicamente en tablas de flujo y una acción asociada a cada una de sus entradas.

Un switch OpenFlow se fundamenta en tres partes: una tabla de flujo con acciones para procesar el flujo de entrada, un camino seguro de datos que permita interactuar los dispositivos de red y con el controlador mediante el protocolo OpenFlow, con el cual el controlador pueda modificar, insertar y eliminar flujos de la tabla de forma proactiva y reactiva, y por último el propio

46 La traducción de direcciones de red o *NAT* (del inglés *Network Address Translation*) es un mecanismo utilizado por routers *IP* para intercambiar paquetes entre dos redes que asignan mutuamente direcciones incompatibles.

switch que envía paquetes entre puertos de acuerdo a como se defina en el controlador.

Cada entrada de flujo puede tener una acción básica como las siguientes:

- Enviar el flujo de paquetes a un puerto establecido
- Encapsular y enviar los flujos al controlador para que este decida qué hacer con él.
- Descartar el flujo por temas de seguridad, para prevenir de ataques.

3.2.1. Tipos de Switches

Hay dos tipos de switches: los que no soportan procesamiento de nivel 2 y nivel 3 y switches OpenFlow-*only*.

Los paquetes interactúan con las tablas de flujo, para esto es necesario que haya al menos una tabla de flujo aunque puede haber más. Cuantas menos entradas contenga la tabla de flujos más simple será el proceso que realice.

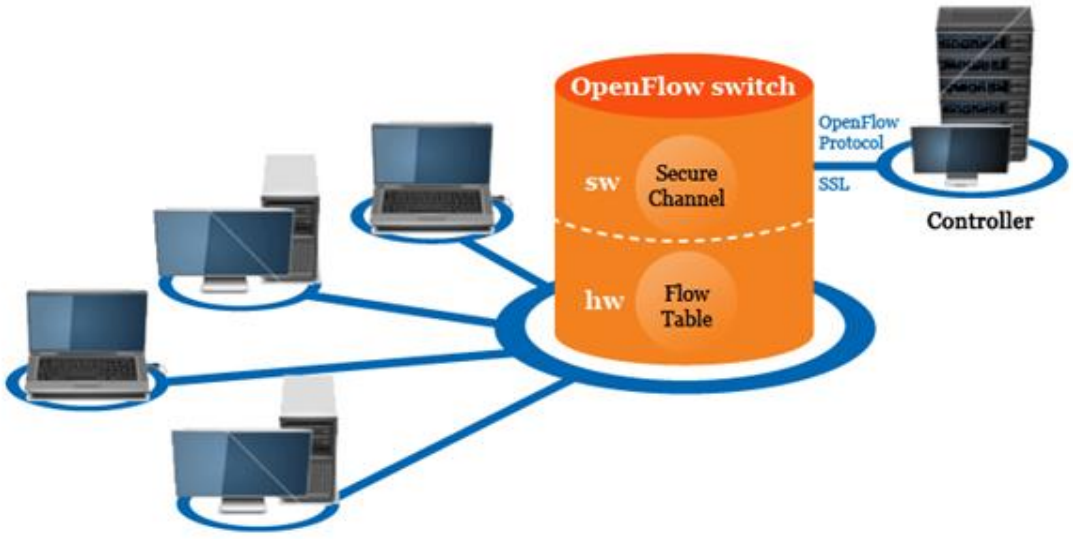

Figura 27: La figura muestra un ejemplo de funcionamiento de un switch OpenFlow.

Los flujos están ampliamente definidos, solo están limitados por la capacidad de una determinada tabla de Flujo. Por ejemplo, un flujo puede ser una

conexión TCP[47], o los paquetes de una determinada dirección MAC[48], los paquetes que vengan de la misma VLAN, o todos los paquetes de un mismo puerto del switch.

Los switches OpenFlow-híbridos son switches que soportan tanto la operación OpenFlow como el encaminamiento *Ethernet* convencional. Las operaciones habituales suelen ser: encaminamiento *Ethernet* de Capa 2, aislamiento de tráfico con VLAN, listas de acceso o QoS. Estos switches deberán proveer un mecanismo de clasificación fuera de OpenFlow que enrute el tráfico, o bien ser procesado por OpenFlow.

El protocolo OpenFlow permite que el switch sea controlado por varios controladores para incrementar el rendimiento del sistema.

3.2.2. Tabla de Flujos

Las tablas de flujo tienen el siguiente formato:

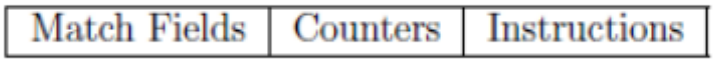

Figura 28: Campos de una tabla de flujos.

- Campos coincidentes (*Match Fields*): a través de estos campos clave que son puerto y cabecera, se diferencia entre los diferentes flujos de entrada.
- Contadores (*counters*): se actualizan cuando encuentran entradas coincidentes.
- Instrucciones (*Instructions*): modifican el conjunto de acciones.

---

47 *Transmission Control Protocol* (*TCP*) o Protocolo de Control de Transmisión, es uno de los protocolos fundamentales en Internet.

48 En las redes de computadoras, la dirección *MAC* (siglas en inglés *de Media Access Control*; en español "control de acceso al medio") es un identificador de 48 bits (6 bloques hexadecimales) que corresponde de forma única a una tarjeta o dispositivo de red. Se conoce también como dirección física, y es única para cada dispositivo.

El proceso de *matching* (coincidencias) comienza cuando el paquete llega al switch. Este comienza haciendo una búsqueda en la primera tabla, con los resultados de esa búsqueda realiza otras búsquedas en las otras tablas de flujo. Se procede a buscar según el orden de prioridad en busca de campos coincidentes, si se encuentra en la primera tabla se ejecutan las opciones asociadas, si no se procede a buscar en las siguientes tablas. En caso de no producirse el *match*, el paquete es enviado al controlador para que decida qué hacer. [17]

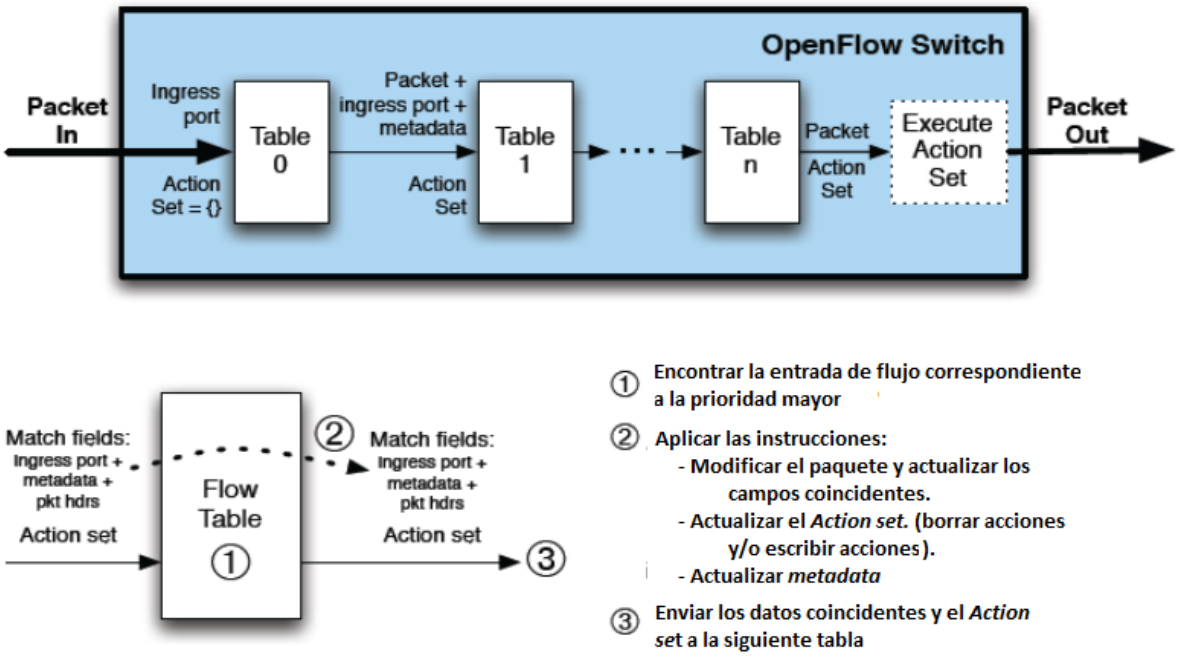

Figura 29: Múltiples tablas de flujo OpenFlow.

3.3. Mensajes OpenFlow

OpenFlow utiliza los mensajes para indicar las normas que deben seguir los switches. Se engloban en tres grupos: del controlador al switch, asíncrono y asimétrico.

3.3.1. Mensajes del controlador al switch

Se inician por medio del controlador y se utilizan para:

- Administrar directamente o inspeccionar el estado del switch.

- Borrar o modificar definiciones de flujo.
- Obtener información de los contadores.
- Enviar paquetes de vuelta al switch para que estos sean procesados después de que se añada un nuevo flujo.

No necesitan respuesta por parte del switch necesariamente. Los mensajes son, en este orden: *Features* (funciones de consulta), *modify-state* (añaden / borran / modifican entradas), *read-state y packet-out*.

3.3.2. Mensajes asíncronos

Estos mensajes se deben a alguna alteración en el estado del switch. Pueden ser generados sin necesidad de que previamente el controlador haya generado un mensaje. Se envían por alguna de estas razones:

- Enviar al controlador algún paquete que no coincida con los flujos.
- Informar de que algún mensaje ha sido eliminado por que ha expirado su TTL[49].
- Informar al controlador de si ha habido cambios en el estado de algún puerto.

Estos mensajes se denominan: *packet-in, flow-removed y port-status*.

3.3.3. Mensajes simétricos

Este tipo de mensajes se envían en cualquier dirección sin una solicitud previa. Los motivos por los que se generan son:

- Comenzar iniciar una sincronización.
- Determinar la latencia de la comunicación entre el switch y el controlador y verificar la conexión.
- Mensajes de prueba para futuras conexiones

---

49 Tiempo de Vida o *Time To Live* (TTL) es un concepto usado en redes de computadores para indicar por cuántos nodos puede pasar un paquete antes de ser descartado por la red o devuelto a su origen.

El nombre de los mensajes que se envían son: *hello, echo (request/reply) y experimenter*. [18]

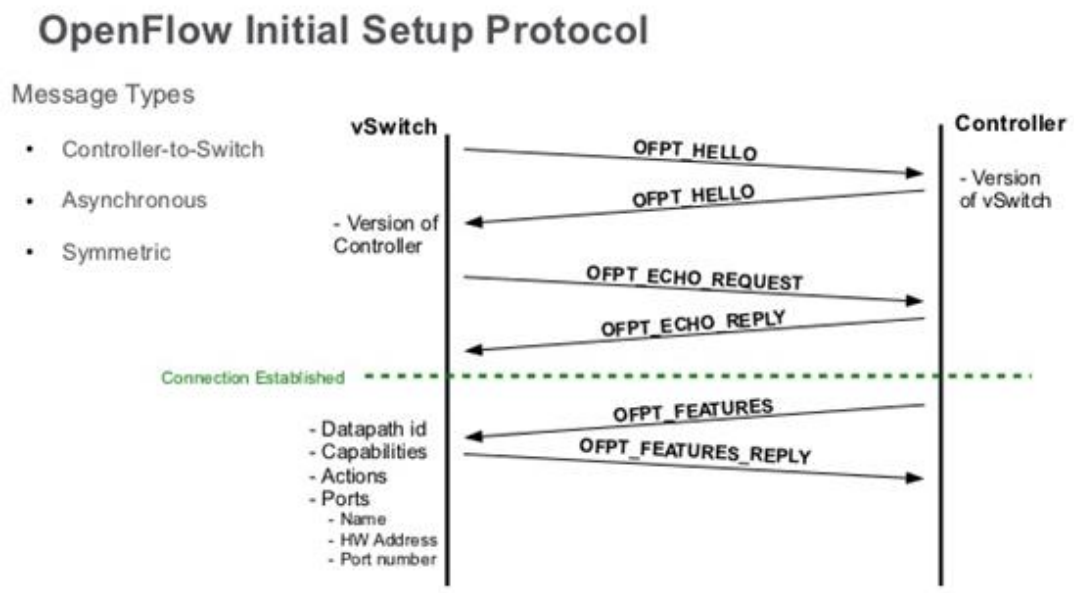

Figura 30: Configuración inicial de mensajes OpenFlow.

En la figura 30 se observa el intercambio de mensajes al comenzar la configuración inicial. Primero, se envían primero mensajes del tipo *hello* para iniciar la sincronización. Como se ve, el que inicia los mensajes es el controlador. Después, para verificar la conexión y determinar la latencia que existe, aparecen los mensajes *echo*. En este caso el que envía el mensaje primero es el switch, ya que los mensajes de este tipo pueden iniciarse en cualquier dirección al igual que los anteriores. Por último están los mensajes del tipo *features* y es aquí donde comienza la función de consulta del controlador al switch, además estos siempre se inician por parte del controlador.

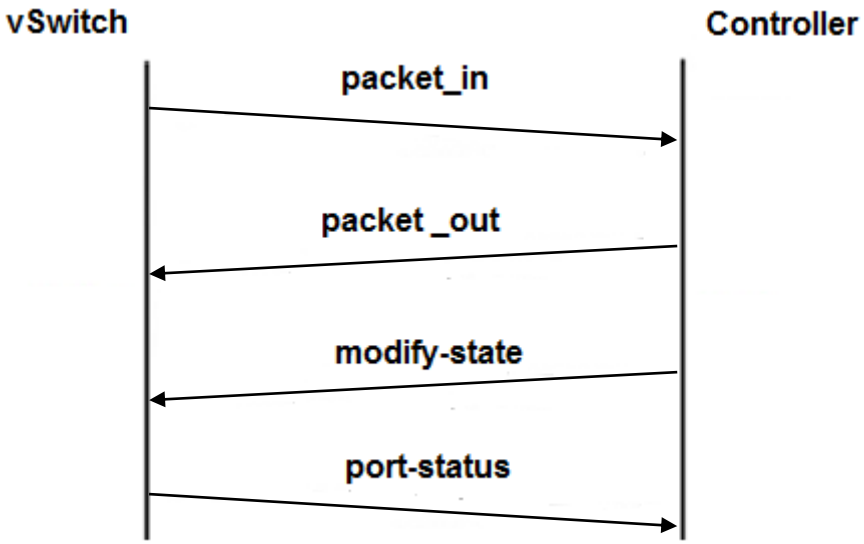

Figura 31: Intercambio de mensajes OpenFlow

La figura 31 muestra un ejemplo de cómo puede continuar el intercambio de mensajes. El switch envía al controlador un mensaje *packet_in* porque un paquete no coincide con los flujos. Este le contesta con un *packet_ou*t de vuelta al switch para añadir un nuevo flujo. Más tarde, el controlador envía un *modify_estate* para añadir/borrar/modificar alguna entrada de la tabla de flujo. Por último, el switch contesta con el mensaje *port_status* para indicar que ha habido modificaciones en su tabla.

Capítulo 4. SDN OPTICAS

4.1. Introducción

Los nuevos y emergentes casos de uso, tales como la interconexión de los centros de datos geográficamente remotos, están atrayendo la atención sobre la necesidad del aprovisionamiento de extremo a extremo de servicios de conectividad que abarcan múltiples y heterogéneos dominios de red. Esta heterogeneidad no sólo se debe a la transmisión de datos y la tecnología de conmutación (el llamado plano de datos), sino también para el plano de control implantado, que puede ser utilizado dentro de cada dominio para automatizar la configuración y la recuperación de dichos servicios, de forma dinámica. La elección de un plano de control, se ve afectada por factores tales como la disponibilidad, la madurez, la preferencia del operador, y la capacidad de satisfacer una lista de requisitos funcionales. Dada la evolución actual alrededor de OpenFlow y SDN, junto con la necesidad de tener en cuenta las implementaciones existentes basadas en GMPLS, el problema de la heterogeneidad de interconexión del plano de control necesita ser resuelto. La solución adoptada debe tratar por igual los temas específicos de las redes multi-dominio, como la visibilidad de la topología de un dominio limitado, dadas las limitaciones de escalabilidad y confidencialidad que los caracterizan. [19]

Por otra parte, ya que en los últimos años se ha avanzado mucho en la modulación óptica y los formatos de codificación; además, hay ensayos prácticos que muestran que esto está mejorando (llegando a Tbps). A pesar de esos avances, hay un gran desequilibrio entre el crecimiento en la capacidad de transmisión y los ingresos en las redes. Esta descompensación se debe a:

- Una gestión de red ineficiente.
- Limitaciones en la capacidad de red para soportar las características del ancho de banda bajo demanda.

Aunque estos desequilibrios los sufren todas las redes ópticas en general, se ha observado que SDN tiene una similitud con el procesamiento de la señal

digital DSP. Esto se debe a que SDN requiere una plataforma DSP mejorada. [20]

Se propone una extensión de SDN y OpenFlow a redes ópticas. OpenFlow está considerado como una tecnología prometedora para el plano de control en las redes IP de conmutación de paquetes, así como la conmutación de longitud de onda en redes ópticas. [21]

Para este capítulo, se van a estudiar siete propuestas de la literatura para el uso de SDN en redes ópticas. En cada una de ellas se introducirá brevemente la propuesta para después desarrollarla.

4.2. Extender los principios SDN para incluirlos en el transporte óptico

El artículo de Steven Gringeri, *et al.* [22], analiza los beneficios y desafíos de la extensión de conceptos SDN a varias arquitecturas de red de transporte. Las implementaciones del plano de control son más complejas, ya que deben tener en cuenta las limitaciones físicas, incluyendo la accesibilidad óptica de la señal, la disponibilidad del ancho de banda, la granularidad…

El objetivo a largo plazo es aplicar los conceptos SDN, a través de redes de múltiples proveedores y múltiples capas, con el fin de apoyar una estructura de control unificado.

4.2.1. Aplicaciones y características de servicio

Si una aplicación cumple los requisitos de conectividad, ancho de banda, calidad de servicio, y resistencia significará que funciona correctamente. Este apartado analiza dos aplicaciones de las redes de transporte que pueden hacer uso de un paradigma SDN: Cálculo de ruta de conexión y configuración, y el transporte de VPN. La red establece rutas de comunicación entre los puntos finales de la aplicación. Estas rutas, a su vez, deben cumplir los requisitos de

ancho de banda, calidad de servicios asociados al *jitter*[50], pérdida de paquetes y retardos en la aplicación.

El objetivo de los requerimientos en este tipo de redes, es una optimización de rendimiento de las aplicaciones y utilización de recursos para aumentar la eficiencia de la red. La asignación dinámica de los recursos, puede ocasionar que cuando una aplicación no utiliza los recursos asignados, estos sean reasignados a otra aplicación o aplicaciones con el fin de aumentar la eficiencia.

Otra aplicación de la capa de transporte es el ranurado de la red, con lo que se establecen las rutas con limitaciones de ancho de banda, retardos y jitter dedicados a determinados clientes para formar una VPN.

4.2.2. Arquitecturas de red ópticas

Los principios de las SDN se pueden aplicar a diferentes arquitecturas de red de transporte óptico, pero las características de las redes subyacentes determinan la ejecución de las aplicaciones.

Las arquitecturas ópticas se pueden construir para conmutar fibras o puertos, además es posible que contengan múltiples canales en un amplio espectro (>5 THz de ancho de banda). La conmutación óptica provoca trastornos que pueden afectar a la calidad de transmisión y no se pueden asegurar todas las rutas debido a impedimentos ópticos.

50 Se denomina *jitter* (término inglés para fluctuación) a la variabilidad temporal durante el envío de señales digitales, una ligera desviación de la exactitud de la señal de reloj.

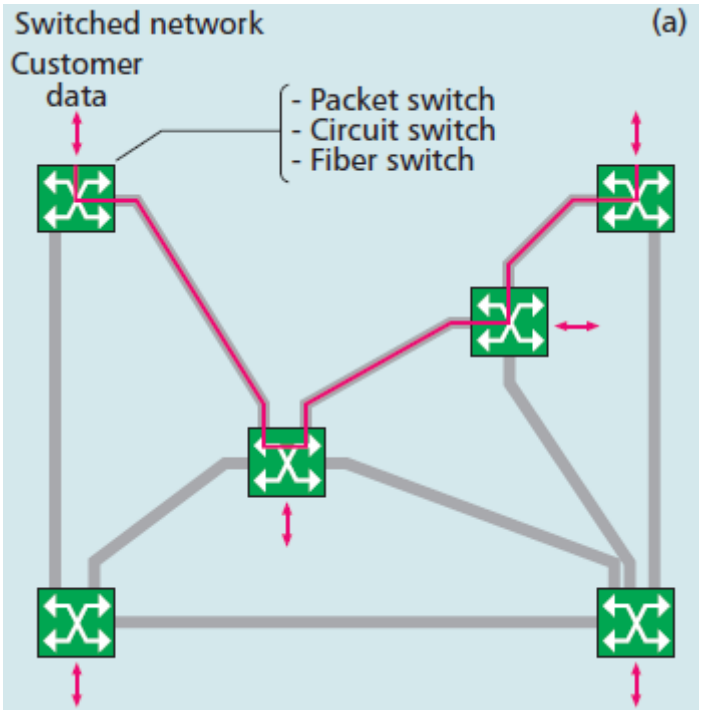

Figura 32: Red conmutada

La figura 32 muestra una red conmutada genérica donde la granularidad de conmutación puede ser de paquetes, de ranura de tiempo, o de fibra.

La conmutación de longitud de onda se implementa normalmente como un multiplexor que puede agregar o eliminar longitudes de onda a través de un nodo. Hay varias estructuras que pueden realizar esto, desde un simple multiplexor fijo, a una estructura capaz de añadir o eliminar longitudes de onda.

La arquitectura de la figura 33, permite la configuración automática de una longitud de onda entre puntos finales- pero la estructura de control generalmente es lenta. A su vez, El alcance del circuito está limitado por el rendimiento óptico de los transceptores y las características físicas de la red.

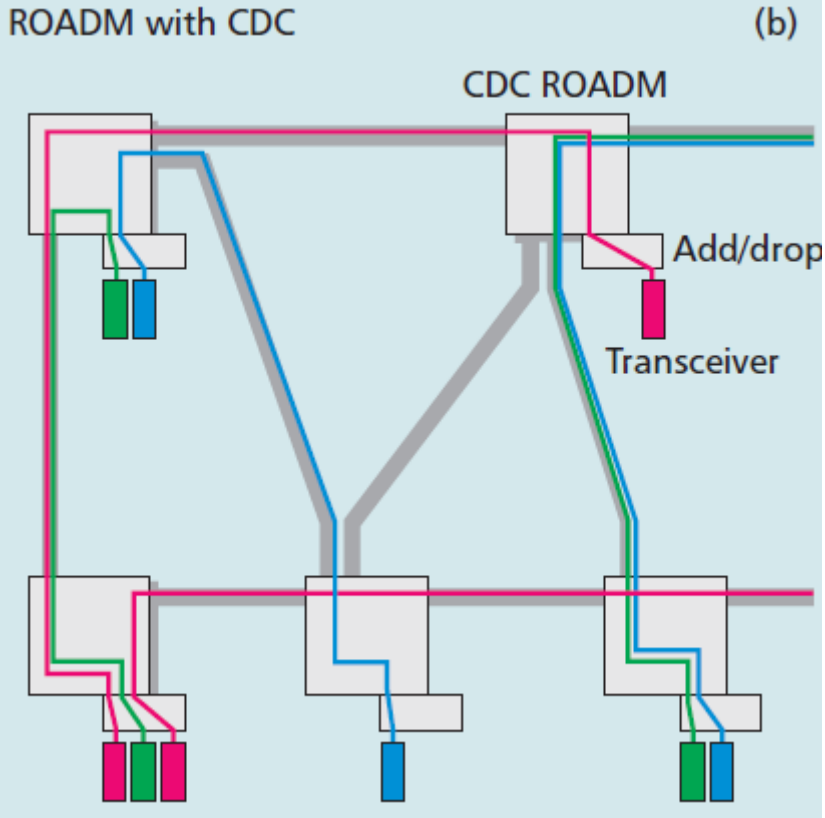

Figura 33: Red basada en CDC ROADM

Tradicionalmente los transceptores tenían una tarea fija, basada en los componentes ópticos. Hoy en día, con la nueva generación de transceptores coherentes, es posible diseñar un hardware flexible para apoyar el equilibrio entre parámetros como el alcance óptico, tasa de bits y eficiencia espectral bajo el control de un software.

Las redes por conmutación de longitud de onda se pueden hacer más dinámicas. Esto es posible gracias a la adicción de capacidad de ajuste de longitud de onda en el transmisor o receptor, con la finalidad de implementar conmutación de sub-longitud de onda. Esta sub-longitud de onda permite el intercambio de longitudes de onda en el dominio del tiempo de manera que el mismo transmisor o receptor puedan comunicarse con múltiples nodos. La capacidad de ajuste, además permite que la red se reconfigure basándose en las demandas de tráfico.

Una de las ventajas que encontramos en estas arquitecturas es que los recursos no son fijos entre dos puntos finales, pero pueden ser reutilizados

dinámicamente según la demanda de tráfico. La selección de estos recursos puede hacerse utilizando un filtro sintonizable o sintonizando el oscilador local en un receptor coherente. El rendimiento global se determina mediante la suma del filtro o el tiempo de sintonización del oscilador local más el tiempo que tarda el receptor en bloquear la señal entrante

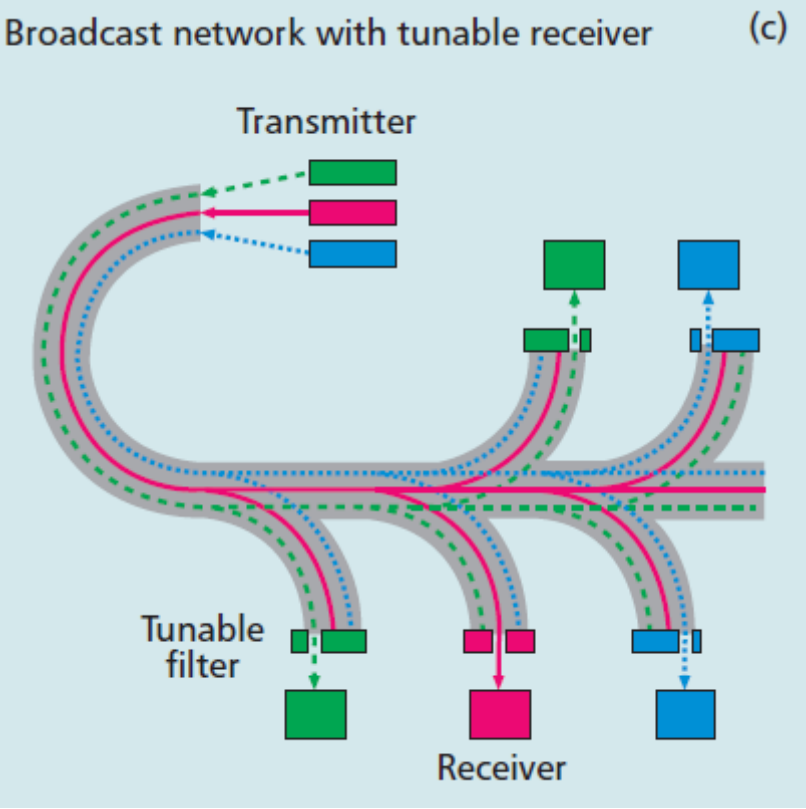

Figura 34: Red *broadcast* con receptor sintonizable.

La figura 34 muestra una red de conmutación de sub-longitud de onda. Esta, utiliza la emisión óptica en una topología de árbol. Esta emisión se hace con la intención de transmitir a cada destino para seleccionar la longitud de onda deseada por un receptor sintonizable.

Se trata de un proceso útil para topologías en anillo de varios cientos de kilómetros de diámetro. Además, se requiere un algoritmo de planificación de hardware o detección de portadora, para evitar que múltiples transmisores emitan simultáneamente al mismo destino (en la misma longitud de onda). La desventaja de diseño es maximizar la utilización del enlace mientras se mantiene la probabilidad de colisión muy baja.

Estas arquitecturas permiten la configuración de conexión rápida entre cualquier punto final. El problema está en que el número de dispositivos finales y la topología de red se ven afectadas por el alcance óptico del transceptor.

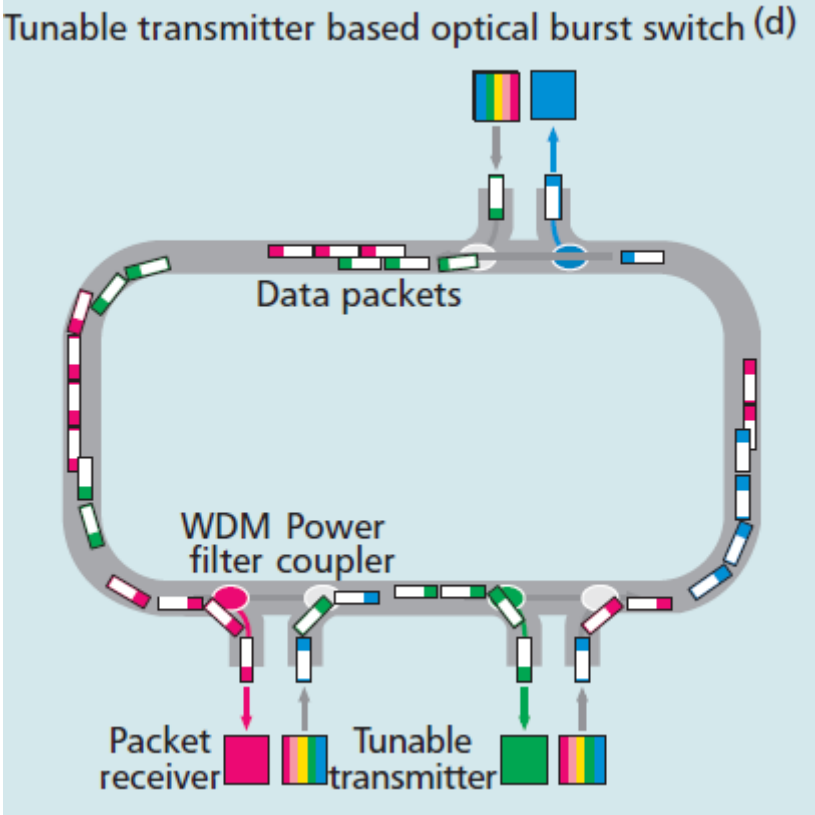

Figura 35: Transmisor sintonizable basado en *burst switches* ópticos.

La Figura 35 muestra una red de conmutación de sub-longitud de onda. En este caso, se basa en un enfoque de conmutación de ráfaga transmisor sintonizable.

4.3. ¿Qué es más adecuado para el Control sobre redes ópticas a gran escala, GMPLS u OpenFlow?

Este artículo de Yongli Zhao, *et al.* [23], se expone que principalmente hay dos arquitecturas de control para redes ópticas multi-dominio. La primera está

basada en GMPLS y PCE (*Path Computation Element*) y la segunda se basa en OpenFlow.

GMPLS (*General Multi-Protocol Label Switching*) se ha desplegado en las redes ópticas actuales con la finalidad de unificar el plano de control. Sin embargo, debido a razones comerciales, no se ha utilizado ampliamente.

Por su parte OpenFlow, ha sido diseñado para diversas aplicaciones en redes ópticas como la comunicación interna entre data centers o la ruta de control del espectro óptico elástico ranurado.

A continuación, vamos a ver estas dos propuestas:

- **Arquitectura basada en el control GMPLS y PCE.**

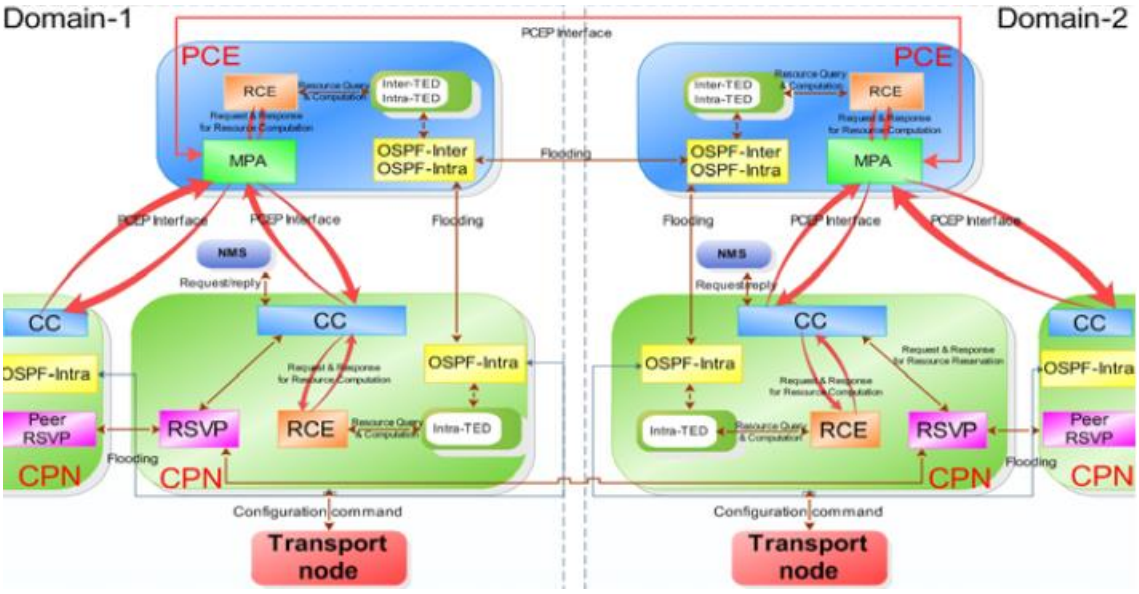

Figura 36: Procedimiento de aprovisionamiento *lightpath*.

Procedimiento de establecimiento de un *lightpath* en la figura 36:

1. CC transfiere la petición a PCE.
2. PCE1 calcula la secuencia de dominio.
3. PCE1 calcula la ruta local de dominio.
4. PCE1 devuelve el resultado a CC.
5. CC desencadena la señalización.
6. PCE2 finaliza los pasos 3, 4 y 5 en el segundo dominio.

7. El último PCE realiza las mismas acciones.

Como se observa en la figura 36, un dominio se define como: cualquier colección de elementos de red dentro de una esfera común de gestión de direcciones o la responsabilidad de cálculo de ruta. Debido a esto, los dominios se pueden clasificar en dominios administrativos o de enrutamiento, contando cada uno de ellos con un PCE en el plano de control. Tanto el PCE como los nodos del plano de control, contienen recursos RCE (*Resource Computation Element*) para completar el cálculo de la ruta de servicio.

Toda la información de topología y la ingeniería de tráfico se sincronizan a través del protocolo OSPF-TE (*Open Shortest Path First with Traffic Engineering*). Por otra parte, está MPA (*Message Policy Analyzer*) que es el lanzador de PCE, y puede interpretar los diferentes tipos de mensajes y las políticas entregados desde CPN u otro PCE a través de protocolos de comunicación, tales como PCEP (*PCE communication protocol*). También es muy importante el CC (*Connection Controller*), este controlador se encarga del mantenimiento y la gestión de diferentes conexiones y comunicaciones entre los diferentes módulos.

Por último, el RSVP-TE (*ReSerVation Protocol with Traffic Engineering*) se utiliza para la reserva de recursos y la configuración de la ruta a lo largo de la LSP de extremo a extremo.

- **Arquitectura basada en el control OpenFlow**

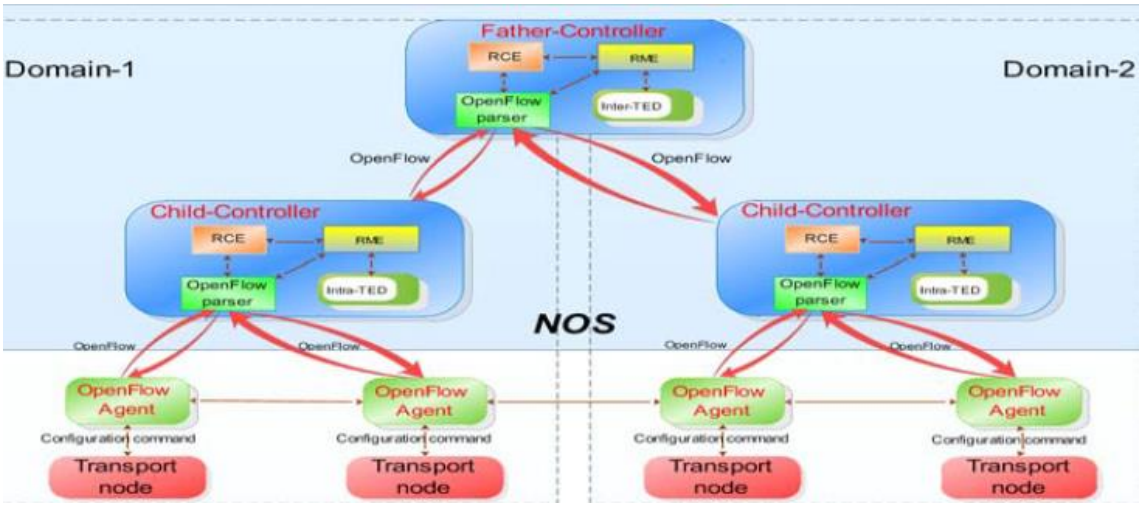

Figura 37: Arquitectura basada en el control OpenFlow

Procedimiento de establecimiento de un *lightpath* en la figura 37:

1. El primer controlador hijo recibe la petición del nodo o cliente
2. El primer controlador hijo determina si el destino es el dominio local
3. Este transfiere la petición a un controlador padre de otro dominio
4. El controlador padre aumenta la secuencia de dominio y lo reenvía a los controladores hijo.
5. El controlador hijo termina el cálculo de ruta y reserva de recursos a lo largo de la secuencia de dominio paralelo.

Como se muestra en la figura 37, este control es más simple que el anterior, también el procedimiento de aprovisionamiento del *lightpath* es más sencillo. No existen nodos del plano de control, por lo que en su lugar, OpenFlow cuenta con todos los nodos de transporte.

Hay dos tipos de controladores en la arquitectura, el controlador padre y el hijo. Cada dominio cuenta con un controlador hijo para el cálculo de la ruta local, y para la asignación de recursos y reservas. Todos estos controladores hijo, están conectados con el controlador padre a través del protocolo OpenFlow. La función de los dos controladores es similar, consiste en analizar OpenFlow, RCE (*Resource Computation Element*),

RME (*Resource Management Element*) y TED (*Traffic Engineering Database*). La diferencia es que el RCE de este protocolo, no solo puede completar el cálculo de ruta, sino también completar la asignación de recursos y reservas.

4.4. Organización de SDN OpenFlow y GMPLS Flexi-Grid con PCE jerárquico con tolerancia a fallos.

En el artículo de Ramon Casellas, *et al.* [19], se presenta un modelo de arquitectura (figura 38) que está compuesto por islas OpenFlow interconectadas entre sí, y a su vez la red de transporte está controlada por GMPLS. También se debe tener en cuenta que, incluso cuando las redes están controladas por una sola entidad administrativa, también pueden ser segmentadas por razones técnicas o de escalabilidad. Se da por hecho, que la interconexión entre dominios se realiza mediante "enlaces fronterizos" (dos dispositivos, uno por cada dominio, están interconectados por un vínculo común) en lugar de "nodos frontera" (un solo elemento de red pertenece a ambos dominios, por lo que un determinado nodo no es administrado y gestionado por dos entidades distintas). Como consecuencia de esto, un solo nodo se une a una sola tecnología de plano de control.

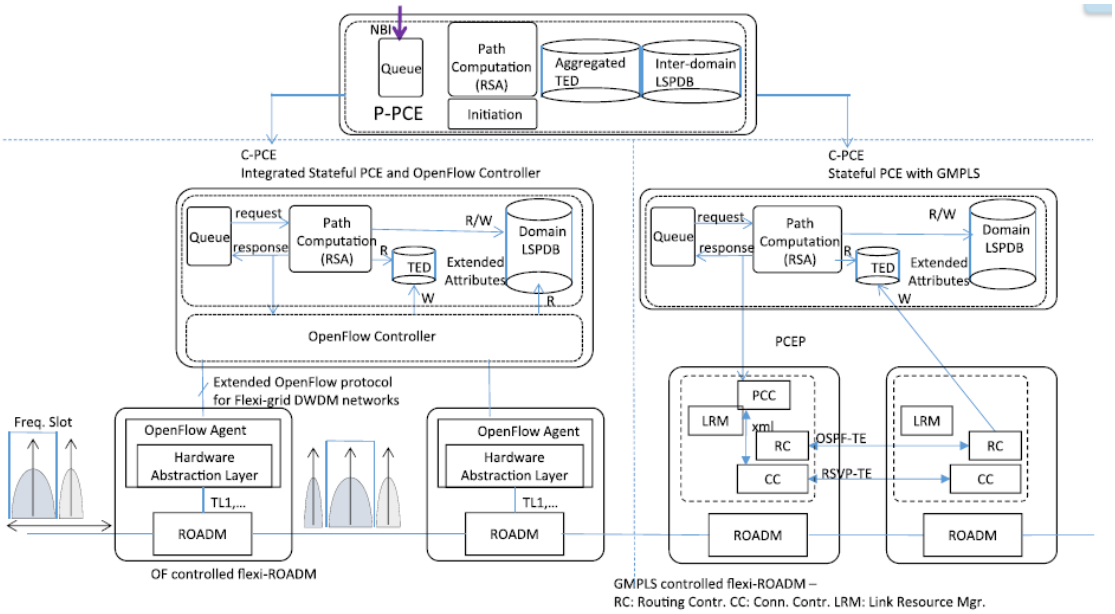

Figura 38: Diagrama de bloques de la arquitectura.

Este tipo de redes multi-dominio se caracterizan porque ninguna entidad tiene visibilidad de la topología completa, afectando así a la optimización y el uso eficiente de los recursos. En esta arquitectura, el PPCE (*par*ent PCE o PCE padre) organiza el suministro de los servicios con los identificadores generalizados, cada CPCE (*The child PCE*) actúa como middleware para cada capa de control del dominio. Cada CPCE controla du dominio, ya sea con un controlador OpenFlow, o delegando a un plano de control GMPLS/PCE. Este último caso, ocurre especialmente en la red óptica central, de alta capacidad y recorrido. Todos los CPEPs tienen el PCEP (*Path Computation Element Communications Protocol*) como su interfaz hacia el norte (NBI). PCEP también es NBI para GMPLS, y se utiliza para el aprovisionamiento, *rerouting* y *reporting*. OpenFlow se utiliza como la interfaz hacia el sur (SBI) para CPCEs en estos dominios.

4.5. SDN y OpenFlow para acceso óptico dinámico Flex-Grid y redes de agregación

El artículo de Neda Cvijetic *et al*. [20] destaca SDN y OpenFlow para las redes ópticas.

Actualmente, en las redes ópticas se requiere de intervención manual tanto entre diferentes segmentos de la capa de red como en la misma capa. Esta intervención incrementa notablemente los costes y ralentiza el aprovisionamiento dinámico de los recursos de rad. La situación puede ser vista como una consecuencia indeseable de la separación del plano de datos y plano de control. Esta característica de las redes SDN que se ha adoptado desde el principio, tiene como resultado grandes gastos en la red y drena la rentabilidad. La desventaja, aunque pueda parecer exclusiva para las redes de transporte ópticas, también lo es para la convergencia de redes que enfrentan fijo y móvil, donde la conectividad de fibra óptica está jugando un papel muy importante.

Para sacar un mayor rendimiento potencial, el controlador SDN debe ser capaz de entender las características claves de la capa óptica. Si no se tiene en cuenta esta consideración, sería complicado poder igualar los recursos de red sin complicar la gestión.

Más allá de estos inconvenientes. Al contrario ocurre en los data center, donde la virtualización del servidor significa hacer que un dispositivo se vea como varias máquinas virtuales. Aqui si se quiere crear un circuito óptico con la implantación de OpenFlow. El reto es enlazar varios dispositivos y hacer que se vean como un switch. Para poder lograr este desafío, se deben tener en cuenta los deterioros de canal y las restricciones de la topología. En consecuencia, la selección del sub-espacio óptimo es la cuestión a resolver y está siendo tratada por los organismos de normalización como la Open Networking Foundation (ONF) y el Optical Internetworking Forum (OIF).

4.6. Habilitación de tecnologías de redes ópticas Flexi-Grid basadas en OpenFlow

A los efectos de la aplicación del software programable para redes ópticas futuras, el artículo de Jiawei Zhang, *et al.* [21], propone un plano de control basado en redes ópticas *Flexi-Grid* OpenFlow. Esto es posible a las características del protocolo como:

- La diferenciación entre las redes de conmutación de circuitos y conmutación de paquetes. OpenFlow originalmente ha sido diseñado para redes de conmutación de paquetes. Así el primer paquete entrante se encapsula y se envía al controlador, este lo devuelve como una entrada de flujo que se añade a la tabla a través del mensaje *Flow_mod* (envío del comando *lighpath setup/release a BVOSs*). Cada switch por el que pasa el paquete repite este proceso hasta que llega al nodo de destino. La entrada de este flujo, es igual que un comando software para activar el switch. Una vez que se hay configurado el *lightpath*, es posible garantizar que el procesamiento de datos siga el camino correcto.
- El ancho de banda variable del switch óptico basado en OpenFlow. En primer lugar, es necesario que el software del switch OpenFlow esté integrado para poder mantener la comunicación entre el controlador y el switch. Más tarde, el switch recibe la orden de *Flow_mod* desde el controlador y lo guarda en su tabla de flujo. Por último, la señal óptica se puede modular a sub-portadoras, con el formato de modulación de acuerdo con el requisito de tráfico, y se pone en marcha con un láser sintonizable. Este proceso lo manipula el controlador, a través el interfaz de control de hardware (HCI, *Hardware Control Interface*).

Para satisfacer el requisito de redes ópticas *Flexi-grid*, se hacen algunas extensiones al protocolo OpenFlow actual:

1. Mensaje *Packet_in*: éste se amplía con dos funciones. Una de ellas es llevar la información de petición *lightpath* que incluye 32 bits, "*Source/Destination addr*", 16 bits "tasa de tráfico" y 32 bits "*ID Path*". Además, también se extiende para devolver un mensaje al controlador cuando BVOs (*Bandwidth-Variable Optical Switch*) completa la acción. 32 bits "*ID Path*" y 8 bits de "*Reason*" se extienden al presentar el ID de un lightpath y el resultado de la acción a este *lightpath* (éxito o fracaso).
2. Mensaje *Flow_mod*. Este se extiende para el envío de configuración del *lightpath* al BVO. Las extensiones correspondientes implican la estructura "*OFP_Match*" y el campo "*Command*".

3. Mensaje *Stats_request/reply*: Se extiende para informar al controlador del estado de flujo óptico. La estructura del "*OFPST_Current_traffic*" es nueva y presenta la tasa de tráfico actual de este *lightpath*.

Por último, se presenta el procedimiento de señalización para la configuración *lightpath* dinámica y el ajuste para el tráfico variable en el tiempo en las redes ópticas *Flexi-Grid*.

Hoy en día prevalece el uso de la variable temporal en el tráfico. Habitualmente en las redes ópticas WDM, se proporciona un canal fijo para el tráfico variable en el tiempo. Esto conduce a la utilización del ancho de banda distinto durante un tiempo de duración diferente. Teniendo en cuenta las ventajas de las redes ópticas *Flexi-Grid*, el tamaño del espectro en el canal óptico se puede proporcionar de manera flexible, según las necesidades de tráfico. El flujo de entrada es actualizado por el controlador para provocar la acción en el switch. Cuando esto ocurre, los BVOs envían la información de ajuste del espectro (*Stats_reply*) al controlador y se genera una nueva entrada de flujo. A continuación, el controlador actualizara la entrada de flujo relativa a al BVO para completar el ajuste del espectro.

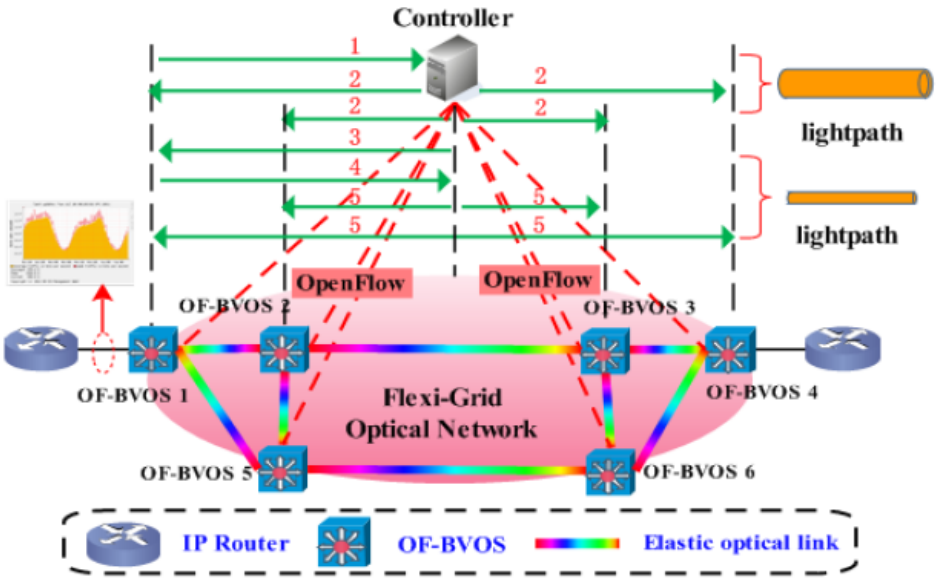

Figura 39: Procedimiento de aprovisionamiento de un *lightpath* elástico.

Como se muestra en la figura 39, el tráfico que entra en las redes ópticas *Flexi-Grid* se presenta como una variable en el tiempo. Para utilizar con eficacia el recurso espectral, podemos ajustar el ancho de banda del *lightpath* de acuerdo con las exigencias del tráfico. El proceso se desarrolla de la siguiente manera:

1. BVOS1 envía la solicitud de conexión al controlador para establecer un *lightpath*.
2. El controlador realiza un algoritmo RSA *(Rivest, Shamir y Adleman)* y devuelve una entrada de flujo al OF-BVOS relacionado a través del mensaje "*Flow_mod*" y se establece el *lightpath*.
3. El controlador consulta al BVOS1 acerca de la tasa de tráfico con el mensaje "*Stats_request*".
4. Cuando la tasa de tráfico cambie, BVOS1 devuelve un mensaje al controlador para informar de la tasa de tráfico actual, a través de un "*Stats_reply*".
5. El controlador envía una nueva entrada de flujo al BVOS para ajustar el tamaño del espectro de *lightpath*, con otro "*Flow_mod*".

4.7. Habilitación de Operaciones SDN ópticas

En este artículo de Channegowda *et al.* [24] se explica como la separación del plano de datos y el plano de control, hace que SDN sea adecuado para soportar múltiples dominios de la red y una gran variedad de tecnologías de transporte. SDN, a través del protocolo OpenFlow, proporciona un nuevo marco para la evolución de la calidad operacional y servicios en la nube. También proporciona una plataforma de plano de control unificado, con la finalidad de integrar los paquetes electrónicos y las redes ópticas.

SDN da apoyo a la interconectividad de recursos IT, tal como la computación en máquina virtual, el almacenamiento con la utilización de transporte óptico y las tecnologías como por ejemplo redes ópticas elásticas. Una consecuencia de estas posibilidades, es la gestión del flujo de tráfico y enrutamiento en

DCs[51]. Cabe destacar también que SDN facilita la aparición de la ingeniería de tráfico programable en DC.

La implementación de SDN en una infraestructura DC multi-tecnologia permite:

- Automatización. Lo hace más eficiente mediante el mapeo de flujos de tráfico. Esto tiene lugar en las capas de transporte de paquetes ópticos y eléctricos en los DCs, independientemente de la tecnología de transporte.
- Aplicación específica y ranurada de IT además de recursos en la red inter e intra DC para crear un DC virtual.

El desarrollo de tecnología óptica SDN basada en la nube conlleva nuevos retos, debido a las diversas características de flujo de tráfico que poseen los servicios en la nube. El plano de control, debe considerar estas características para poder asignar correctamente los recursos de infraestructura a los requerimientos de los usuarios o aplicaciones.

La arquitectura de redes ópticas SDN basada en OpenFlow que presenta Channegowda *et al.*, es adecuada para los servicios de computación en la nube. La arquitectura permite la implementación de redes agiles y elásticas, que puedan adaptarse a los requisitos de aplicación bajo demanda.

Esta arquitectura tiene los siguientes obstáculos:

- Definir un transporte óptico unificado que se pueda generalizar para diferentes tecnologías de transporte ópticas. También debe permitir la compatibilidad con los paquetes eléctricos.
- Diseñar un mecanismo de abstracción que no tenga en cuenta los detalles de la capa óptica. Esta abstracción tiene modelos similares a los sistemas operativos de teléfonos móviles como Android.

---

51 Un Sistema de Control Distribuido o SCD, más conocido por sus siglas en inglés DCS (*Distributed Control System*).

- Tener en cuenta las características de las capas físicas de las tecnologías de transporte óptico.
- Las limitaciones de la tecnología en la asignación del ancho de banda.

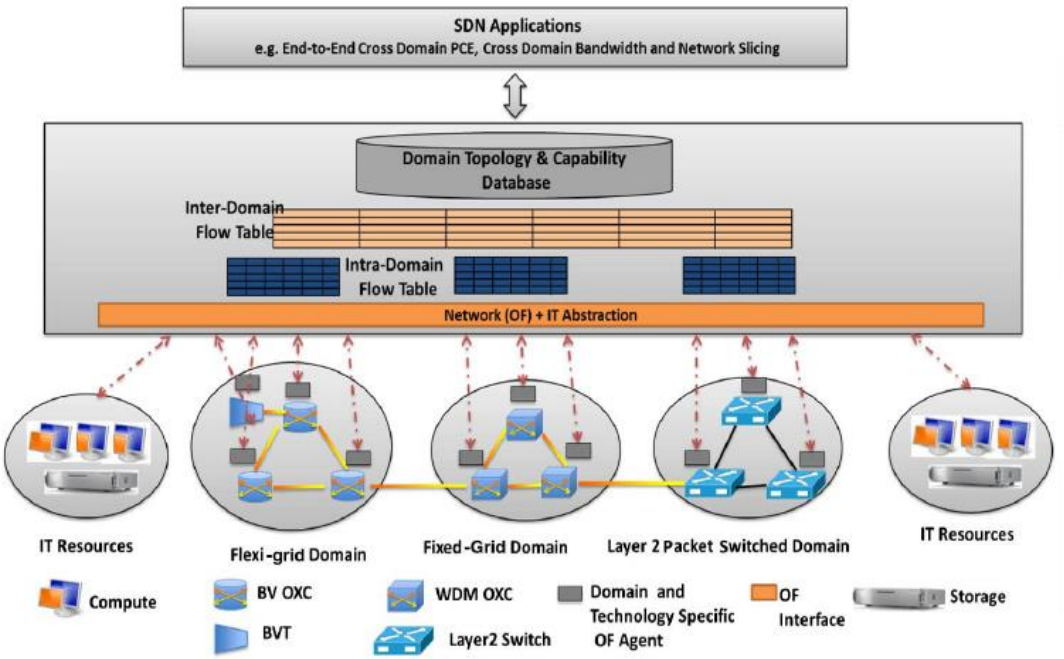

Figura 40: Arquitectura de múltiples capas del plano de control

La figura 40 muestra el diagrama de bloques de la propuesta de una arquitectura de plano de control óptico basado en SDN. Como se ha mencionado anteriormente, el núcleo de esta tecnología es el mecanismo de abstracción, llevado a cabo por un controlador y el protocolo OpenFlow.

4.7.1. Abstracción de Hardware

El principal objetivo, es ocultar los detalles tecnológicos de recursos de la red y habilitar una interfaz programable para la configuración de hardware.

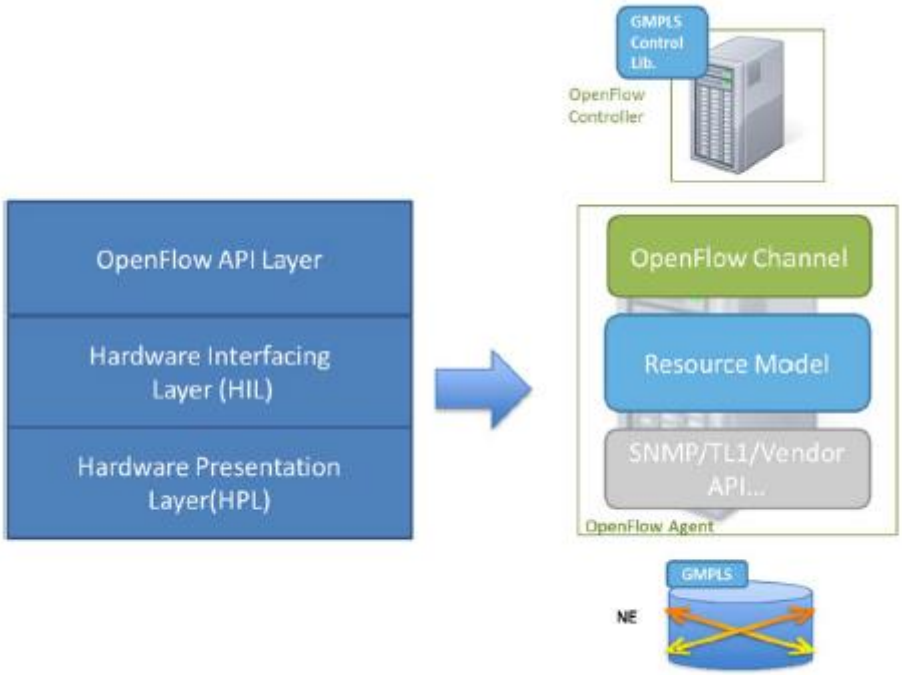

Figura 41: Abstracciones OpenFlow.

En esta figura se muestran tres capas:

1. HPL (capa de presentación hardware). El laminado de alta precisión proporciona todas las capacidades del dispositivo, oculta las desventajas y exporta las funciones del dispositivo y capacidades basadas en un modelo unificado para la capa superior (HIL).
2. HIL (capa de interfaz de hardware). Utiliza las interfaces proporcionadas para construir abstracciones útiles, dejando a un lado la complejidad que aportan los recursos de hardware.
3. OpenFlow API abstrae información proporcionada por la HIL.

La principal diferencia entre HPL y HIL es que el primero expone todas las capacidades disponibles de un dispositivo y el segundo solo las necesarias para la abstracción general basada en flujo.

4.7.2. Extensión OpenFlow

La versión actual de OpenFlow, se concentra principalmente en los dominios de paquetes. Esta versión es la 1.0 y posee extensiones para soportar la

conmutación de circuitos, pero no admite funciones de red ópticos. A esta versión, se le añadió un apéndice para hacer frente al dominio óptico, considerando una red óptica síncrona.

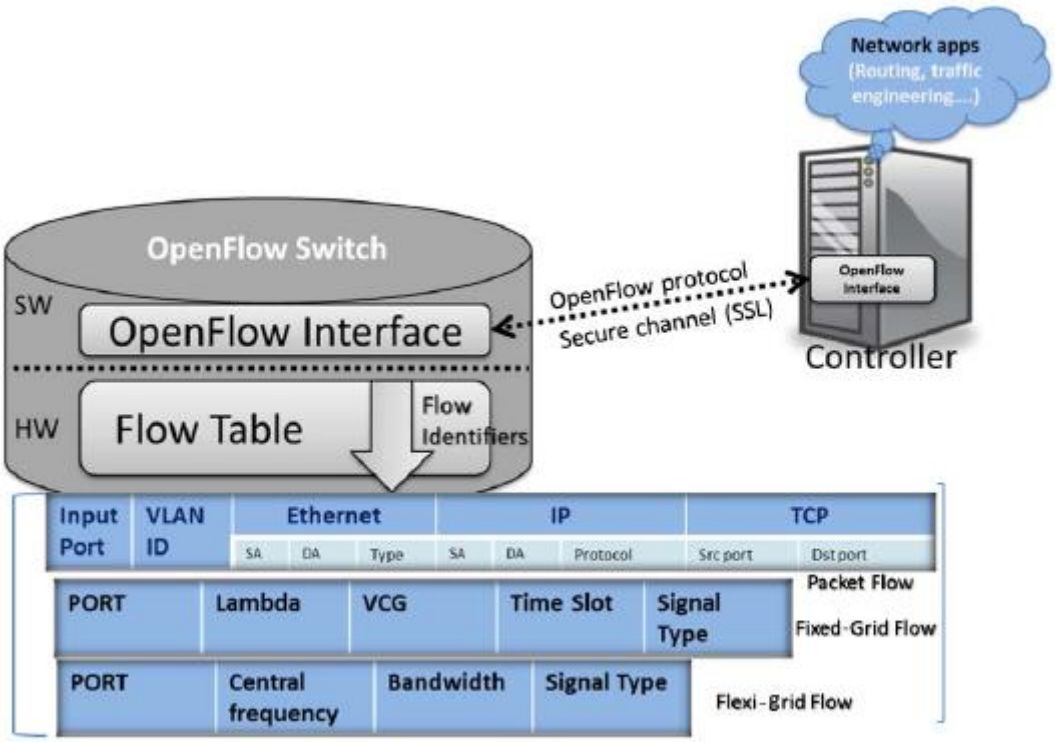

Figura 42: Definiciones de flujo para diferentes dominios tecnológicos.

En la figura 42 se muestra que un flujo óptico puede ser identificado por un identificador de flujo que posee: puerto, longitud de onda o frecuencia central de la portadora óptica, ancho de banda asociado y especificaciones de la capa física.

4.7.3. Aplicación de SDN

Las aplicaciones proporcionan funcionalidades de red aisladas. Estas ofrecen una forma modular de añadir o modificar nuevas funcionalidades.

En la arquitectura que se ha propuesto (figura 40), son los diferentes algoritmos para el enrutamiento de longitud de onda o el espectro, los que se utilizan como aplicaciones. Al mismo tiempo, las nuevas aplicaciones, tienen tareas

como el cálculo de la ruta, la asignación de longitud de onda, evitar bucles, y otras muchas tareas más que resultan críticas en una red de paquetes óptica.

El controlador OpenFlow tiene una API bien definida en la que los algoritmos, o las nuevas aplicaciones, son utilizados para proporcionar múltiples funciones.

En la arquitectura que se propuso anteriormente, el problema principal es la optimización de los recursos para un entorno en la nube dinámico. Esto tiene que ver con los flujos de datos, ya que deben ser creados cuidadosamente para que no haya subutilización, especialmente en el dominio óptico de alta capacidad. Por ejemplo, una alta capacidad de flujo de tráfico de baja latencia podría ser atractivo para el dominio óptico, pero si es en ráfagas cortas entonces conduce a una asignación de recursos ineficiente. La solución a este problema, es desarrollar un equilibrador de carga de aplicaciones que equilibre el flujo de tráfico basado en requisitos de aplicación. Esto se podrá llevar a cabo teniendo en cuenta las aplicaciones de dominio de la tecnología y el ancho de banda.

4.8. Eficiencia energética con control de QoS en redes ópticas dinámicas con SDN

El artículo de Wang *et al.* [25] presenta los algoritmos de enrutamiento basados en la nueva plataforma de plano de control integrado. Pretende la centralización de la estructura del plano de control, lo que permite la elección para las rutas en redes ópticas. Este algoritmo sirve para optimizar el rendimiento energético en toda la red, pero para que sea eficiente debe buscar un equilibrio entre optimización energética y *QoS*. Esto se ha estudiado para los casos de prioridad alta, y satisfactoriamente muestran mejoras en el rendimiento manteniendo la *QoS*.

También, se han considerado varios dominios de red para esta creación de redes energéticamente eficientes. Algunas de las soluciones son, la baja potencia en las horas de inactividad o la utilización de ingeniería de tráfico. Sin embargo, en todas estas soluciones no está considerado el hecho de mantener

la *QoS* al mismo tiempo, y que esto no es una solución válida para lo que se requiere. Por esta misma razón se propone la forma de enrutamiento flexible, utilizando una plataforma con el nuevo plano de control integrado.

Esta nueva plataforma, mejora las operaciones de red tradicionales mediante la comunicación eficiente a través de múltiples dominios de red. Los mecanismos de control de red actuales son separaciones horizontales entre las redes centrales y las de acceso, y las separaciones verticales que son entre las capas de aplicación, transporte y física. Estas separaciones conllevan altos costes operativos y de energía, una baja eficiencia de la red y alta latencia en la ruta de aprovisionamiento.

La aparición de aplicaciones con altos requerimientos de velocidad de datos y bajas demandas de retardo, como son televisión de alta definición o *VoIP*[52]. Propone nuevos desafíos para aumentar el rendimiento, disminuyendo los costes energéticos, mientras se mantiene la *QoS*.

Por supuesto, estos requerimientos no son fáciles de cumplir, se necesita un mecanismo unificado de control de dominio. Para esto, se han considerado conceptos SDN ya que desacopla los planos de control y de datos y permite un plano de control integrado a través de dominios. Un plano de control programable, proporciona flexibilidad en el diseño de nuevas funcionalidades y servicios con independencia de la capa física subyacente.

Aunque el control centralizado exige información de seguimiento para ser llevado a una única entidad, imposición de una latencia y comunicación de sobrecarga. Este control centralizado, es un modelo adecuado de superposición para diseñar el plano de control integrado. En contraste con esto, los mecanismos distribuidos, pueden sufrir inconsistencia en toda la red debido a la latencia y también requieren del intercambio de información para mantener el comportamiento convergente.

---

52 Voz sobre IP, (*VoIP* por sus siglas en inglés, *Voice over IP*, Voz por IP), es un grupo de recursos que hacen posible que la señal de voz viaje a través de Internet empleando un protocolo IP (Protocolo de Internet).

Tradicionalmente, los algoritmos de enrutamiento dinámico como SP (ruta más corta) o LB (equilibrio de carga), solo pueden evaluar un parámetro a la vez. Mediante la aplicación de algoritmos de enrutamiento eficiente de energía, por ejemplo, LC (menor consumo), se pueden lograr ahorros energéticos sensibles sin influir en otro tipo de tráfico. Sin embargo, estos ahorros son limitados, ya que solo uno de estos (consumo energético, retardo o bloqueo) se cumple a la vez.

El nuevo algoritmo, en lugar de utilizar el número de saltos como el único objetivo para el tráfico sensible a retardo, deja el consumo de energía como algo secundario. El MOEA (*Multi-Objective Evolutionary Algorithm*) se utiliza para mejorar la selección de la ruta en un contexto de red dinámica. Este, aunque se requiere cierto tiempo para ejecutarlo, se prueba en un entorno de simulación dinámica, existiendo una tolerancia de retardo asumible al configurar una ruta de red de núcleo óptico.

El plano de control integrado centralizado no sólo permite un intercambio de información más eficiente a través de diferentes dominios, sino también le da la posibilidad de implementar algoritmos más potentes. Todo esto con la finalidad de mejorar la relación entre *QoS* y ahorro de energía que se consigue con la introducción del algoritmo evolutivo multi-objetivo para optimizar este rendimiento.

4.9. Conclusiones

Como conclusión a los trabajos expuestos, se propone un plano de control integrado centralizado para permitir un control unificado de la red. Dicho plano, no sólo permite un intercambio de información más eficiente a través de diferentes dominios, sino también le da la posibilidad de implementar algoritmos más potentes. A su vez, también ofrece un marco para que las aplicaciones soliciten y reciban los servicios de la red a través de un controlador SDN centralizado.

Las conclusiones en cuanto a rendimiento y eficiencia energética, se pueden encontrar diferentes puntos de vista (Tabla 1):

- En el artículo de Jiayuan Wang *et al.* [21], el objetivo es abordar el conflicto entre ahorro de energía y mantenimiento de *QoS*. Propone un algoritmo multi-objetivo para optimizar el rendimiento energético a través de toda la red.
- El artículo de Steven Gringeri, *et al.* [22], concluye analizando que en un plano de control centralizado, el rendimiento se puede mejorar mediante la adicción de potencia de procesamiento y pre-calculando las rutas.
- En el artículo de Neda Cvijetic *et al*. [20] se concluye que para minimizar tanto la latencia como el consumo de energía, la conmutación y el enrutamiento deben tener lugar en la capa de red más baja que sea posible.
- El artículo de Channegowda *et al.* [24], realiza una comparativa de rendimiento entre GMPLS con OpenFlow integrado y OpenFlow independiente. Los términos de la comparativa son los tiempos de configuración de ruta en la que concluye que tiene mayor rendimiento en ese aspecto OpenFlow independiente.
- Los resultados numéricos del artículo de Yongli Zhao, *et al.* [23] muestran que la arquitectura OpenFlow presenta un mejor rendimiento que la arquitectura GMPLS en redes ópticas multi-dominio a gran escala.
- Por último, el artículo de Wang *et al.* [25] presenta los algoritmos de enrutamiento de eficiencia energética basados en una plataforma de plano de control integrado.

Otro punto que comparten algunos de los artículos es la permisividad de redes elásticas (tabla 1):

- En el artículo de Channegowda *et al.* [24], la arquitectura propuesta permite la implementación de *cloud networks* elásticas. Esto quiere decir que pueden adaptarse a los requerimientos de la aplicación sobre demanda.

- El artículo de Yongli Zhao, *et al.* [23], menciona que OpenFlow está diseñado para aplicaciones tales como el control de rutas elásticas en redes ópticas.
- Respecto a la figura 39 en el apartado referente al artículo de Jiawei Zhang, *et al.* [21], muestra un procedimiento de señalización del aprovisionamiento *lightpath* elástico en redes ópticas *Flexi-Grid* basadas en OpenFlow.

Finalmente, en todos los artículos menos en el de Jiawei Zhang, *et al.* [21], se incluye el protocolo OpenFlow como base para unificar el plano de control para una SDN óptica. Se pueden ver las similitudes y diferencias en la tabla 1.

ARTICULO	Rendimiento y eficiencia energética	Permisividad de redes elásticas	Protocolo utilizado
Ramon Casellas, *et al.* [19]			Interconexión de GMPLS y OpenFlow
Neda Cvijetic *et al*. [20]	Para minimizar tanto la latencia como el consumo de energía, la conmutación y enrutamiento debe tener lugar en la capa de red más baja que sea posible		Uso de Openflow en redes ópticas para la creación dinámica del circuito de longitud de onda *Flexi-grid*
Jiawei Zhang, *et al.* [21]	Propone un algoritmo multi-objetivo para optimizar el rendimiento energético a través de toda la red	Procedimiento de señalización del aprovisionamiento *lightpath* elástico en redes ópticas *Flexi-Grid* basadas en OpenFlow	Uso de OpenFlow en el plano de control para redes ópticas Flexi-grid
Steven Gringeri, *et al.* [22]	El rendimiento se puede mejorar mediante la adicción de potencia de procesamiento y pre-calculando las rutas		Uso de OpenFlow en el plano de control para redes ópticas Flexi-grid
Yongli Zhao, *et al.* [23]	la arquitectura OpenFlow presenta un mejor rendimiento que la arquitectura GMPLS en redes ópticas multi-dominio a gran escala	OpenFlow está diseñado para aplicaciones tales como el control de rutas elásticas en redes ópticas	Pruebas de OpenFlow comparándolo con GMPLS
Channegowda *et al.* [24]	OpenFlow independiente tiene mayor rendimiento en términos de los tiempos de configuración de ruta	Permite la implementación de *cloud networks* elásticas	OpenFlow para unificar el plano de control en SDN óptica adaptada a servicios en la nube
Wang *et al.* [25]	Algoritmos de enrutamiento de eficiencia energética basados en una plataforma de plano de control integrado		

Tabla 1: Conclusiones SDN ópticas

Capítulo 5. CONCLUSIONES Y LINEAS FUTURAS

En el presente trabajo fin de grado, se ha expuesto el marco teórico las redes definidas por software o SDN. Cabe destacar, que la estructura de estas redes tiene como principio fundamental la separación del plano de datos y el plano de control. En esta separación, un aspecto fundamental es el protocolo (OpenFlow es el más extendido) ya que hace posible el entendimiento entre dispositivos de red y controlador. Esa separación entre los planos de control y datos, es la principal característica que hace a este tipo de redes más programables y eficientes en relación con las redes tradicionales. Esto las permite adaptarse a los cambios en el tráfico con mucho menor impacto.

Para implementar el cambio, existen varios controladores de código abierto que pueden gestionar las redes definidas por software como por ejemplo: NOX, POX, Beacon, Floodlight o Ryu. Cada uno de ellos tiene unas características distintas, como el lenguaje de programación del controlador, la forma de tratar los flujos o la versión OpenFlow que soportan.

El punto clave de este trabajo, es la inclusión de SDN en las redes ópticas. Para ello, he revisado la literatura sobre estos temas y me he centrado en seis propuestas distintas. De cada propuesta, he analizado cada una de sus características para después sacar las conclusiones oportunas que pueden ser resumidas en la Tabla1.

En la investigación sobre este tema, me he basado en los artículos de Steven Gringeri, *et al. ,* Yongli Zhao, *et al. ,* Ramon Casellas, *et al. ,* Neda Cvijetic *et al. ,* Jiawei Zhang, *et al. ,* Channegowda *et al. ,* y Wang *et al.* De estos artículos, he analizado cada una de sus propuestas detalladamente para después sacar las conclusiones oportunas. Estas conclusiones se ven resumidas en la tabla 1.

Uno de los criterios que he tomado para comparar estos artículos es el rendimiento y eficiencia energética. Aquí, se exponen las conclusiones de los autores para aumentar el rendimiento o los resultados de sus pruebas.

Otro punto interesante es en el que se compara las diferentes formas que tienen estos autores de implementar el protocolo OpenFlow como base para unificar el plano de control para una SDN óptica.

El camino por el que debería seguir este estudio, es la creación de una SDN física para tener resultados reales sobre esta investigación. Analizando los tiempos de retardo, gestión de flujos, procesamiento de datos, conectividad entre dispositivos, aplicación del protocolo OpenFlow…

Un aspecto muy importante seria experimentar con diferentes controladores y comparar su funcionamiento. Realizar un estudio de cual de ellos se adapta mejor a un determinado tipo de topología o cual gestiona mejor las redes con un gran numero de dispositivos. Con este estudio se sacarán conclusiones sobre cual se debería implementar en cada caso de uso, con la finalidad de darle una utilidad al experimento.

Por ultimo, el punto clave en futuras investigaciones es el uso de SDN en redes ópticas. Me parece interesante el establecimiento de OpenFlow en dispositivos ópticos de red, descubrir cual son los requerimientos y describir las implementaciones del protocolo en equipos ópticos comerciales. Para esto, se tendría que evaluar el análisis de rendimiento en la arquitectura a estudio. Por otra parte, probar la eficiencia de SDN en la nube con la intención de comprobar si la red es flexible y dinámica, que es el propósito final de SDN.

Anexo.

DESCRIPCION DE MININET Y SUS FUNCIONALIDADES

Mininet es un emulador de red que emula una serie de dispositivo en un núcleo Linux. Se puede fácilmente interactuar con la red mediante mininet CLI, personalizarlo, o implementarlo en hardware real, mininet es útil para el desarrollo, la enseñanza y la investigación.

Esta herramienta permite crear, personalizar e interactuar con un prototipo de SDN. Una característica muy importante de mininet, es que al simular un entorno realista permite fácilmente el cambio a un prototipo real; además es una solución poco costosa al estar disponible de manera gratuita. Soporta múltiples topologías predefinidas que son fáciles de crear con un solo ejecutar un comando en su API, también permite crear tus propias topologías escritas en Python.

Mininet interactúa con el protocolo OpenFlow para establecer comunicaciones entre el controlador y los switches. [3]

1. Porque utilizar mininet

Características básicas:

- Crea un banco de pruebas. Consiste en simular una red para aplicaciones relacionadas con el protocolo OpenFlow.
- Permite utilizar múltiples topologías. Tantas y tan diversas como el usuario necesite.
- Permite realizar pruebas en topologías complejas. Además no tiene la necesidad de cablear una red física.

- Incluye un CLI. Consiste en una pequeña aplicación que se ejecuta en la consola, sirve para ejecutar todo tipo de pruebas utilizando los comandos propios del programa.
- Incluye una serie de topologías predefinidas para realizar pruebas básicas sin necesidad de crear una topología desde un primer momento.
- Se puede utilizar sin tener que programar. Esto es posible mediante una sencilla y extensible API Python para la creación de redes y la experimentación.
- Mininet responde de manera correcta en la ejecución de pruebas y simula perfectamente una topología física.
- Ejecuta un código real, incluyendo las aplicaciones estándar de red Unix / Linux, así como el núcleo y una pila de red Linux real.

Como consecuencia, el código a desarrollar y probar en mininet, para un controlador OpenFlow, switch, o host, puede correr en un sistema real con muy pocos cambios. Esto significa que un diseño que funciona en mininet generalmente puede pasar directamente a los switches de hardware para el envío de paquetes.

Mininet contiene las características de los mejores simuladores, bancos de pruebas y emuladores:

- Carga a una velocidad rápida en comparación con los sistemas físicos (segundos en lugar de minutos).
- Permite la emulación de un gran número de hosts y switches.
- Proporciona un ancho de banda (generalmente de 2Gbps) en un hardware adecuado al programa.
- Puede instalarse fácilmente. Bien mediante una máquina virtual, o mediante instalación nativa en el propio equipo
- Es un software gratuito y se puede configurar y reiniciar fácilmente.
- Se interactúa fácilmente, pudiendo escribir y ver los diferentes procesos que está realizando. [26]

2. Trabajar con mininet

Una vez esté instalado el programa se puede comenzar a realizar distintas pruebas, utilizando los diferentes comandos que nos ofrece mininet.

2.1. Crear topologías

Lo primero que tenemos que hacer para lanzar mininet es crear una topología para después poder someterla a diferentes pruebas. Puede ser una topología básica, predefinida o personalizada.

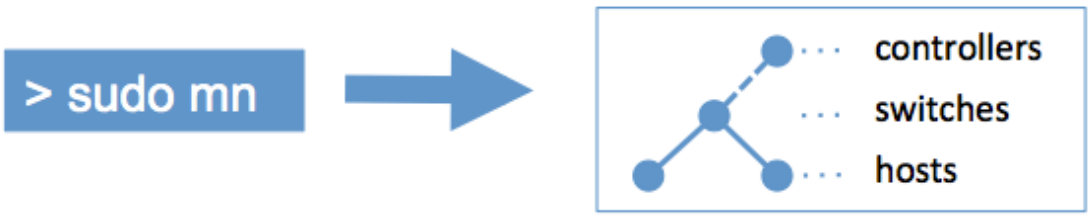

Figura 43: Crear topología en mininet.

2.1.1. Topología básica

Es el emblema del programa mininet, consiste en un controlador que se comunica con un switch y dos hosts que están conectados a ese switch. El comando para lanzarla es *sudo mn*.

Se necesitan permisos de superusuario[53] para iniciar mininet por eso el "*sudo*" y *mn* para abrir el programa. No se pone nada más porque es una abreviatura del comando original de la topología preinstalada como *minimal (sudo mn --topo minimal).*

2.1.2. Topologías predefinidas

Hay distintos tipos de topologías predefinidas en la propia instalación de mininet. Para iniciar una topología se debe usar la opción *--topo* que hace

53 En sistemas operativos del tipo Unix, *root* es el nombre convencional de la cuenta de usuario que posee todos los derechos en todos los modos (mono o multi usuario). *root* es también llamado superusuario.

referencia a la topología que vamos a escribir a continuación. El comando seria *sudo mn –topo*…

- *Linear*: esta topología genera una red con un número definido de switches y de hosts conectados linealmente como vemos en la imagen:

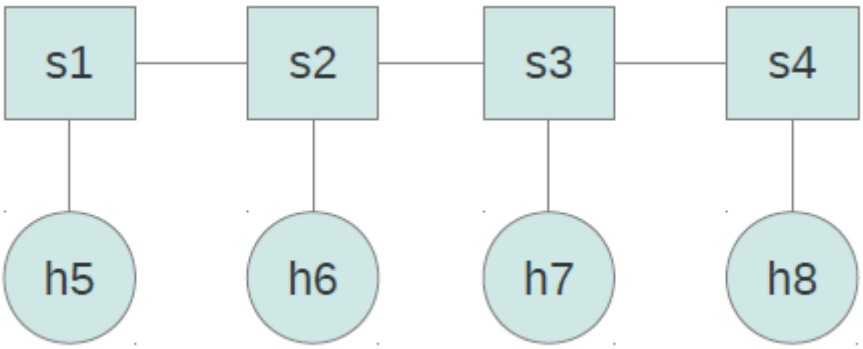

Figura 44: Topología linear en mininet.

Todos los switches están conectados a su vez al controlador, que es el que genera las órdenes del flujo.

El número de switches, que será el mismo que el de hosts, para esta topología se especifica de la siguiente manera: *sudo mn --topo linear,M*. Donde M es el número de hosts y switches que tendremos en la topología.

- Single: esta topología construye una red con un solo switch conectado a un número definido de host como vemos:

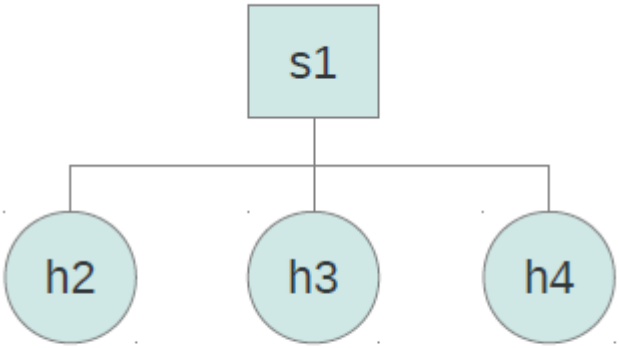

Figura 45: Topología single en mininet.

Este switch, también está conectado al controlador y de la misma manera que el anterior, podemos definir el número de host como: *sudo mn --topo single,M.* Con M el número de host que van a formar parte de la topología.

- *Tree*: Es una topología como su propio nombre indica en forma de árbol. El comando para su creación es el siguiente: sudo mn --topo tree,N,M.

  Para esta topología se tienen dos variables: N que es el nivel de profundidad del árbol que se desea tener, y M es un parámetro que indica el número de host conectados a cada switch de último nivel.

  En el siguiente ejemplo (figura 47) N=2 y M=2:

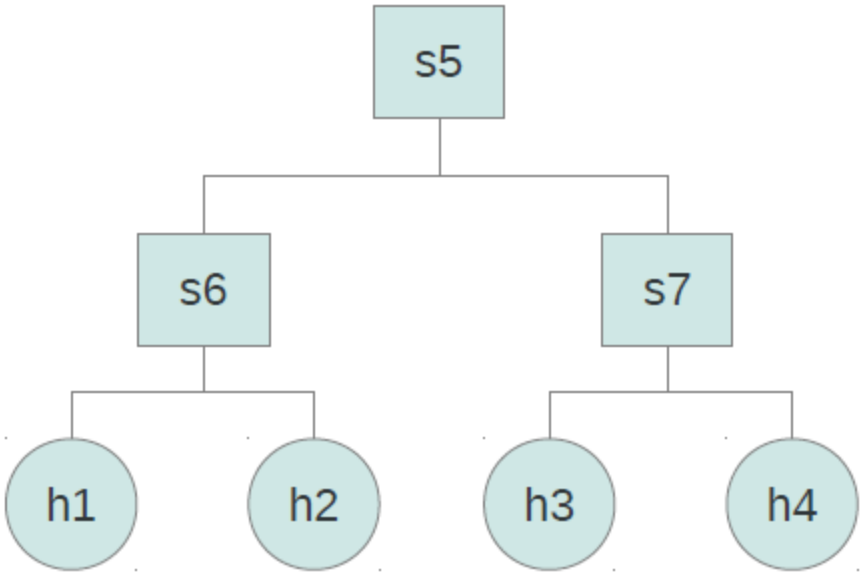

Figura 46: Topología *tree* en mininiet

2.1.3. Topologías personalizadas

Si ninguna de las topologías anteriormente escritas satisface nuestros requerimientos, es necesario hacer un *script*[54] en lenguaje de programación Python.

Ejemplo:

54 Un *script*, archivo de órdenes, archivo de procesamiento por lotes o guión. Es un programa usualmente simple, que por lo regular se almacena en un archivo de texto plano.

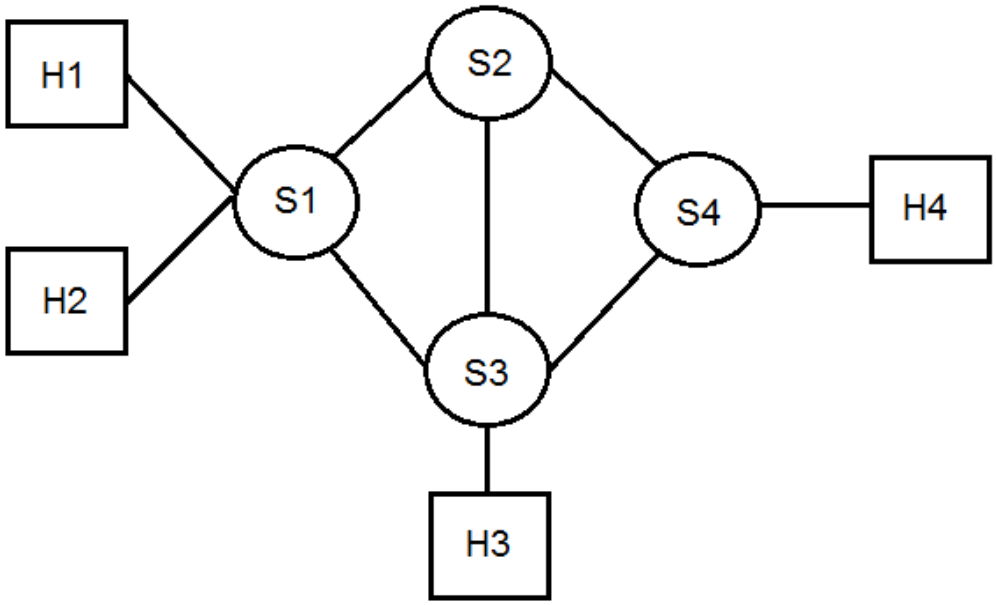

Figura 47: Ejemplo de topología personalizada en mininet.

El código python que genera este ejemplo es el siguiente:

```
"""Custom topology example """


from mininet.topo import Topo


class MyTopo( Topo ):

    "Ejemplo TFG Ruben de Paz Villarroel"


    def __init__( self ):

        "Create custom topo."


        # Initialize topology

        Topo.__init__( self )
```

```
        # Add hosts and switches
        Host1 = self.addHost( 'h1' )
        Host2 = self.addHost( 'h2' )
             Host3 = self.addHost( 'h3' )
             Host4 = self.addHost( 'h4' )
        Switch1 = self.addSwitch( 's1' )
        Switch2 = self.addSwitch( 's2' )
             Switch3 = self.addSwitch( 's3' )
             Switch4 = self.addSwitch( 's4' )

        # Add links
        self.addLink( Host1, Switch1 )
        self.addLink( Host2, Switch1 )
        self.addLink( Switch1, Switch2 )
             self.addLink( Switch1, Switch3 )
             self.addLink( Switch2, Switch3 )
             self.addLink( Switch4, Switch2 )
             self.addLink( Switch4, Switch3 )
        self.addLink( Switch3, Host3 )
             self.addLink( Switch4, Host4 )

topos = { 'mytopo': ( lambda: MyTopo() ) }
```

2.2. Comandos de mininet

Los comandos pueden estar escritos en diferentes ámbitos

- $ Comandos de Linux escritos en la *Shell*[55] *prompt*[56]. Se puede utilizar *$sudo mn –h* para ver las opciones de inicio que ofrece mininet.
- # precede a comandos de Linux que están escritos en el directorio raíz.
- Mininet> es la interfaz CLI de mininet en la que se escribirán los comandos propios del programa.

Se describen a continuación algunos de los comandos más útiles en mininet, se suelen utilizar en una simulación de red.

- *Help*: describe los comandos que podemos utilizar en mininet
- *Nodes*: muestra por pantalla los diferentes nodos de la red. Bien sea hosts, switches o controlador.
- *Net*: muestra las conexiones entre los nodos.
- *Dump*: vuelca la información de los nodos como la IP, PID, lo…

Cuando previamente escribes el nombre de un nodo, el comando actuara sobre este y no sobre toda la red.

- *Ifconfig*: si utilizas este comando sobre un nodo veras toda su información, así como la del bucle local *loopbak* (ej: *h1 ifconfig*).
- *Ps*: para ver el conjunto de procesos bien de un host o de un switch (ej: *s1 ps*).

Para testear la conexión entre dos host utilizamos los siguientes comandos

- *Ping*. Se generan mensajes ARP[57] desde un host para la dirección MAC de la segunda, lo que provoca un mensaje *packet_in* para el controlador.

---

55 En informática, el término *shell* o Intérprete de órdenes se emplea para referirse a aquellos programas que proveen una interfaz de usuario para acceder a los servicios del sistema operativo

56 Se llama *prompt* al carácter o conjunto de caracteres que se muestran en una línea de comandos para indicar que está a la espera de órdenes

Este genera entonces un mensaje *packet_out* para inundar a otros puertos en ese switch. El segundo host vera la petición y enviara una respuesta de difusión que va al controlador y lo envía al primero.

Ahora el primer host conoce la dirección del segundo y puede mandar el ping a través de una petición ICMP[58]. Esta petición junto con la respuesta van al controlador y se convierten en una entrada de flujo (ej: *h1 ping h2*).

Una vez que se conozca la dirección el mensaje de ping tardara menos en recibirse.

- *Pingall*: es una forma de hacer ping a todos los pares de host que haya en la red.
- Se pueden ejecutar pruebas sin necesidad de entrar en el CLI de mininet con la opción --test. Por ejemplo:
    - *Sudo mn --test pingpair*: este comando crea una topología (mínima en este caso) y realiza una prueba de todos los pares de host. Más tarde borra la topología creada mostrando los resultados.
    - *Sudo mn --test iperf*: este comando crea un servidor *iperf* en un host, y un cliente en el otro para analizar el ancho de banda logrado. Nos muestra el resultado en Mb/s y posteriormente elimina la topología creada.

Para una mayor depuración se pueden utilizar ventanas *xterm*[59]. Para abrir una ventana *xterm* se utiliza la opción -x. también se pueden abrir este tipo de ventanas una vez estemos en el CLI utilizando el comando *xterm* seguido del

---

57 ARP (del inglés *Address Resolution Protocol* o, en español, Protocolo de resolución de direcciones) es un protocolo de la capa de enlace de datos responsable de encontrar la dirección hardware (Ethernet MAC) que corresponde a una determinada dirección IP.

58 El Protocolo de Mensajes de Control de Internet o ICMP (por sus siglas en inglés de *Internet Control Message Protocol*) es el sub protocolo de control y notificación de errores del Protocolo de Internet (IP)

59 *Xterm* es un emulador de terminal para el sistema de ventanas *X Window System*

dispositivo que se quiera abrir. Este comando resulta útil para abrirlo en los diferentes host ya que en los switch es equivalente a un terminal regular.

También es muy útil para la gestión de flujos en los diferentes switches con el comando *dpctl add-flows* (añadir) y dependiendo del puerto de escucha lo introduciremos en un switch o en otro. [26]

2.2.1. Comando dpctl

Este comando merece una mención especial ya que permite la crear y manipular flujos desde cualquier host o switch que entienda OpenFlow.

Para ver en la API de mininet todas las opciones que permite utilizar este programa se ejecuta *dpctl –h*.

Dpctl envía un mensaje de petición de características al switch, el cual le contesta con un mensaje de respuesta con las características requeridas.

La opción más interesante de este comando es la del tipo *add-flow*, que permite insertar un flujo con las opciones que se deseen. [15]

La sintaxis del comando *add-flow* es la siguiente:

```
#dpctl add-flow [protocolo]:[ip_conmutador]:[puerto] nw_proto=[protocolo_filtrado],nw_src=[ip_origen],nw_dst=[ip_destino],dl_src=[mac_origen],dl_dst=[mac_destino],tp_src=[puerto_origen],tp_dst=[puerto_destino],in_port=[#puerto_entrada],idle_timeout=[tiempo_expiración_inactividad],hard_timeout[tiempo_expiración],dl_vlan=[VLAN],priority[prioridad],actions=[output:#puerto_salida]/[normal]/[flood]/[enqueue:puerto:id]/[all]/[controller]/[mod_vlan_id:id]
```

Los diferentes parámetros vienen explicados a continuación:

- [Protocolo]: es el protocolo mediante el cual se realizará la comunicación. Puede ser TCP, SSL o Unix (SSL si se requiere una conexión segura). Unix para trabajar en una maquina local o TCP para una transmisión sin cifrado.
- [ip_conmutador]: la dirección IP asignada al switch.
- [Puerto]: es el puerto por el que el controlador escucha los mensajes OpenFlow. (Después de varias pruebas la conclusión es que el puerto por el

que comienza a escuchar OpenFlow es el 6634 y ese se le asigna al primer switch, a los siguientes se les va asignando números de puerto ascendentes 6635,6636…)

- [Protocolo_filtrado]: filtra los paquetes del protocolo que se indica.
- [ip_origen] e [ip_destino]: como su propio nombre indica, aquí se especifican las direcciones IP de origen y destino del flujo
- [mac_origen] y [mac_destino]: las direcciones físicas MAC del origen y destino.
- [puerto_origen] y [puerto_destino]: identificadores del puerto TCP o UDP.
- [#puerto_entrada]: es el puerto por el que ingresan los mensajes. Normalmente en mininet se van asignando de manera creciente (1, 2, 3…).
- [tiempo_expiración_inactividad]: se refiere al tiempo que se mantiene el flujo en la tabla si no se registra actividad. Para que el flujo este de forma permanente se le asigna el valor 0.
- [VLAN]: identificador de VLAN.
- [Prioridad]: define la prioridad del VLAN.

Opciones de tratamiento del paquete, denominadas *actions*:

  - [output:#puerto_salida]: el puerto de salida al que se refiere el flujo.
  - [normal]: el paquete se comporta de manera normal.
  - [flood]: inunda todos los puertos con el flujo menos el que tenga la opción deshabilitada (*no flood*).
  - [enqueue:puerto:id]: el paquete se pone en cola. Se establece un puerto y un identificador.
  - [all]: envía el paquete a todos los puertos (incluso a los que tienen la opción *flood* deshabilitada).
  - [controller]: envía el paquete al controlador para que este decida cómo tratarlo.
  - [mod_vlan_id:id]: modifica el id de VLAN.

Después para ver los flujos que hay en los diferentes switches se utiliza el comando *dump-flows*, lo más cómodo es abrir un *xterm* previamente y finalmente introducir el comando:

```
#dpctl dump-flows tcp:<ip controlador:puerto>
```

Con todos estos parámetros, se personaliza detalladamente el flujo que se desea introducir para que realice las opciones con las que nosotros le hayamos configurado. Los parámetros que resultan más útiles son (aparte de los de *actions* con los que podemos dirigir el mensaje a la salida que deseemos), los de *timeout* con los que indicamos al flujo el tiempo que puede permanecer en la tabla, bien sea por inactividad o porque se le ha agotado el tiempo.

Una vez establecidos los flujos, estos se pueden modificar con el comando *mod-port*, que tiene la siguiente sintaxis:

```
#dpctl mod-port [protocolo]:[ip_controlador]:[puerto] [puerto_fisico] [estado]
```

Dónde:

- [Puerto_físico]: es el puerto por el cual se identifica el switch.
- [estado]: puede ser *down*, si se desea apagar el puerto, up si se lo desea encender, *flood* cuando se desea una inundación de mensajes y *noflood* cuando se desea deshabilitar el puerto para las inundaciones de flujo.

Un estado de puerto apagado es representado por el identificador 0x1, mientras que el estado noflood se representa por 0x10 en la configuración. [7]

Capítulo 6. BIBLIOGRAFIA

[1]. Nadeau, T. D., & Gray, K. SDN: Software Defined Networks. Beijing, Cambridge, Farnham, Köln, Sebastopol, Tokyo. O'Reilly Media. (2013).

[2].Open Networking Foundation. https://www.opennetworking.org/.

[3]. Velásquez Vargas, W. A. "Emulación de una red definida por software utilizando MiniNet". Escuela Técnica Superior de Ingenieros en Telecomunicaciones (ETSIT - UPM).

[4]. Figuerola, N. "SDN" – Redes definidas por Software. Fuentes consultadas: DatacenterDynamics - CIO - Search Data Center - DICE

[5]. Villarubia, C. "La evolucion que neceditaba la red se llama software definded networks". Datacenter Dinamics, 26-27. (2013).

[6]. Extending SDN across carriers' networks. Intune Networks. (2012).

[7]. Hidalgo, D. A. "Diseño e implementación de una aplicación de red bajo la arquitectura SDN". PONTIFICIA UNIVERSIDAD JAVERIANA. (2014).

[8]. Diego Kreutz, M. I. Software-Defined Networking: A Comprehensive Survey. (2014).

[9]. A.METZLER. Ten Things to Look for in an SDN Controller. www.necam.com. (2013).

[10]. NOXRepo.org. (s.f.). NOX. Obtenido de http://www.noxrepo.org/nox/about-nox/

[11]. NOXRepo.org. (s.f.). POX. Obtenido de http://www.noxrepo.org/pox/about-pox/

[12]. Erickson, D. (s.f.). The Beacon OpenFlow Controller. Stanford University.

[13]. Project Floodlight. Obtenido de Floodlight.: http://www.projectfloodlight.org/floodlight/, (2015).

[14]. Floodlight Controller. Obtenido de JIRA: https://floodlight.atlassian.net/wiki/display/floodlightcontroller/Floodlight+Documentation, (2012).

[15]. GitHub. Obtenido de Dpctl Documentation: https://github.com/, (Enero de 2013).

[16]. Team. RYU SDN Framework (Vol. Using OpenFlow 1.3). (2014)

[17]. Galué, V. A. “ESTUDIO DEL ESTANDAR OPENFLOW MEDIANTE CASO PRÁCTICO DE UTILIZACIÓN CON LA HERRAMIENTA VNX”. Universidad Politécnica de Madrid. (2012).

[18]. Fernández, M. A. “Fundamentos de Open Networking. Soluciones Open Flow. JUNIPER Networks”.

[19]. Ramon Casellas, Raül Muñoz, Ricardo Martínez, Ricard Vilalta, Lei Liu, Takehiro Tsuritani, Itsuro Morita, Víctor López, Oscar González de Dios y Juan Pedro Fernández-Palacios. “SDN Orchestration of OpenFlow and GMPLS Flexi-Grid Networks With a Stateful Hierarchical PCE”. A106 J. OPT. COMMUN. NETW. VOL. 7, NO. 1. 2015.

[20]. Neda Cvijetic, Akihiro Tanaka, Philip N. Ji, Karthik Sethuraman, Shuji Murakami, and Ting Wang. “SDN and OpenFlow for Dynamic Flex-Grid Optical Access and Aggregation Networks”. JOURNAL OF LIGHTWAVE TECHNOLOGY, VOL. 32, NO. 4 p 864, FEBRUARY 15, 2014.

[21]. Jiawei Zhang, Jie Zhang, Yongli Zhao, Hui Yang, Xiaosong Yu, Lei Wang y Xihua Fu. “Experimental demonstration of OpenFlow-based control plane for elastic lightpath provisioning in Flexi-Grid optical networks”. State Key Laboratory of Information Photonics and Optical Communications, Beijing University of Posts and Telecommunications, Department of Researching, ZTE Corporation. Optical Society of America, 2012.

[22]. S. Gringeri, N. Bitar y T.J. Xia, “Extending software defined network principles to include optical transport,” IEEE Communications Magazine, Vol. 51. No. 3. pp. 32-30, Marzo 2013.

[23]. Yongli Zhao, Jie Zhang, Hui Yang y Yiming Yu. “Which Is More Suitable for the Control over Large Scale Optical Networks, GMPLS or OpenFlow?”. State key Laboratory of Information Photonics and Optical Communications, Beijing University of Posts and Telecommunications. OFC/NFOEC Technical Digest ©, 2013.

[24]. Mayur Channegowda, Reza Nejabati y Dimitra Simeonidou “Enabling Software-Defined Optical Network Operations”, J. OPT. COMMUN. NETW./VOL. 5, NO. 10/OCTOBER 2013.

[25]. Jiayuan Wang , Xin Chen , Chris Phillips y Ying Yan. “Energy efficiency with QoS control in dynamic optical networks with SDN enabled integrated control plane”, Computer Networks, vol. 78 (2015), p57–67.

[26]. Team Mininet. Retrieved from http://mininet.org/walkthrough/#part-1-everyday-mininet-usage , (2015).