La virtualización de las redes

PID_00265732

Víctor Huertas García

Tiempo mínimo de dedicación recomendado: 4 horas

Universitat Oberta de Catalunya

Víctor Huertas García

Ingeniero en Telecomunicaciones por la Universitat Politècnica de Catalunya. Actualmente trabaja como ingeniero de *networking* y experto en NGN/IMS en el departamento de Equipos de Comunicación en la multinacional Indra Sistemas. Ha participado en numerosos proyectos de la ESA (Agencia Europea del Espacio) de investigación sobre la aplicación de la tecnología IP en redes satélite. Recientemente ha participado en proyectos de integración de IMS en las redes satélite para conseguir la convergencia con redes terrestres.

El encargo y la creación de este recurso de aprendizaje UOC han sido coordinados por el profesor: Victor Garcia Font (2019)

Segunda edición: septiembre 2019
Autoría: Víctor Huertas García

Av. Tibidabo, 39-43, 08035 Barcelona
Realización editorial: FUOC

Índice

Introducción

Tal y como el paradigma de las redes 5G ya ha introducido, la flexibilización total de las redes de telecomunicaciones busca el objetivo de conseguir una red multiservicio y con capacidad para cumplir con los requisitos de QoS que los casos de uso propuestos para 5G imponen.

Dicha flexibilidad pasa inexorablemente por la virtualización de las redes donde se tiene un nodo central que aglutina el control, la inteligencia y conocimiento de la arquitectura de toda la red. Es lo que llamamos SDN o *Software Defined Networks*.

En definitiva, es como volver a un modelo que ya se ha vivido años atrás, en concreto en los años 70 del siglo pasado cuando las redes de telecomunicaciones de datos acababan de empezar. En esa época el mundo de las computadoras estaba todo centralizado en un nodo al cual se conectaban múltiples terminales con capacidades computacionales muy limitadas. La gente usaba el sistema central para ejecutar sus tareas de computación cotidianas. A finales de los 80 poco a poco el modelo fue evolucionando a un sistema distribuido y la inteligencia se fue desplazando desde el nodo central hacia los terminales remotos.

Pero como ya hemos dicho, con la explosión del *cloud computing*, estamos volviendo a un modelo de centralidad. El *cloud* hará todas las funciones de computación, *networking*, almacenamiento de información y seguridad sin necesidad de tener infraestructura local.

En este módulo nos adentraremos en la descripción de las SDN (*Software Defined Networks*) y el NFV (*Network Functions Virtualization*).

Objetivos

Los contenidos de este módulo han de permitir a los estudiantes los objetivos siguientes:

1. Comprender la influencia de los servicios 5G y sus características en QoS en el impulso de la virtualización de redes.

2. Los beneficios obtenidos como resultado del uso de tecnología SDN (*Software Defined Network*) y NFV (*Network Function Virtualization*).

3. Asimilar y comprender el porqué del cambio de paradigma en la arquitectura funcional de los elementos de una red de telecomunicaciones en un entorno SDN.

4. Aprender qué es un SDN Controller y qué función tiene en el desacoplo entre la capa de datos y la de control.

5. Conocer los bloques en los que se compone un ecosistema NFV según la especificación de la ETSI.

6. Comprender el funcionamiento interno del elemento más importante del NFV: NFV MANO.

7. Comprender cómo se complementan la tecnología SDN y la de NFV.

1. *Software Defined Networks* (SDN)

En las siguientes secciones abordaremos la descripción a alto nivel del cambio de paradigma que las SDN introducen. Sin embargo, antes de empezar creemos importante entender por qué dicha arquitectura de referencia ha sufrido este cambio.

1.1. ¿Por qué vamos hacia la SDN?

El mundo de la provisión de servicios multimedia, tal y como las NGN han introducido, tienden a convertir las redes de transporte en redes multiservicio, donde una sola infraestructura de comunicaciones, independientemente de la tecnología, pueda usarse para ofrecer el máximo número de servicios. Y para ofrecer dichos servicios con unos requerimientos muy heterogéneos de QoS se necesita conseguir una total integración de las redes de transporte.

Las entidades que especifican las redes de telefonía 5G ya nos están dando pistas de cuál es la máxima en la especificación del nuevo modelo de referencia: **las redes han de ser flexibles**. Flexibles para soportar cualquier servicio. Flexibles para poder fragmentarse en dominios administrativos completamente independientes. Flexibles para poder **adaptarse rápidamente** a cambios en los requerimientos a nivel de modelo de negocio (con nuevos servicios, nuevos perfiles, incrementos en el número de usuarios, etc.).

En definitiva, las redes han de permitir que los administradores e ingenieros de redes y operadores puedan adaptar su red a estos cambios. Las ***Software Defined Networks***, siguiendo un modelo centralizado donde la capa de control y la capa de datos están completamente separados, pueden ofrecer la solución.

Elementos de procesado de paquetes en SDN

Los elementos que procesan paquetes en redes SDN deben soportar la comunicación con el elemento de control, llamado SDN Controller, a través de un protocolo específico que debe ser soportado por dicho elemento. Estos elementos se les suele llamar *bare-metal* para recalcar su casi nulo poder de decisión en el procesado y encaminado de paquetes.

¿Pero un modelo distribuido como el que siempre ha existido hasta ahora no puede ofrecer lo mismo?

Efectivamente, el modelo clásico donde se usan protocolos estandarizados cuya capa de control se ubica en cada uno de los elementos de la red puede ofrecer una solución a este reto. Pero tiene principalmente dos dificultades:

- **Limita la escalabilidad:** en una red distribuida, incrementar la capacidad de la red implica que la capa de control de los nuevos elementos añadidos se coordine con los ya existentes no estando exento de impacto en el resto de sistema.

- **Limita la operatividad:** relacionado con el anterior, el hecho de tener que configurar y gestionar cada elemento por separado (cada uno con su

propia interfaz de monitorización y control) hace de la operatividad de la red algo complejo y propenso a fallos.

¿Qué beneficios aporta pasar a un modelo de red centralizado como el que se propone en las *Software Defined Networks*?

El hecho de tener desagregados la parte de control y la parte de procesado de transporte permite que con un solo *software* de control puedes gestionar remotamente infinidad de equipos que lo único que van a hacer es obedecer las configuraciones que reciben desde dicho *software*. Con esto, el problema de la **escalabilidad** de la red estaría solventado y a la vez tendrías una **red completamente integrada**.

Conlleva también **permitir a aplicaciones de terceros controlar la red entera** sin necesidad de que esta conozca la topología física real de la red. De hecho, en SDN todo se puede considerar una aplicación.

Permite también lo que llaman el ***service chaining***, donde en tiempo real podemos reconfigurar partes de la red para poder concatenar varios servicios con el objetivo de flexibilizar su uso.

Finalmente, ofrece la **integración de redes cableadas e inalámbricas** de manera transparente.

Así pues, en las siguientes secciones explicaremos con más detalle qué son las SDNs empezando por la descripción del cambio de paradigma y centrándonos luego en la arquitectura de referencia junto a los principales actores que la definen, de los cuales resaltaremos el SDN Controller.

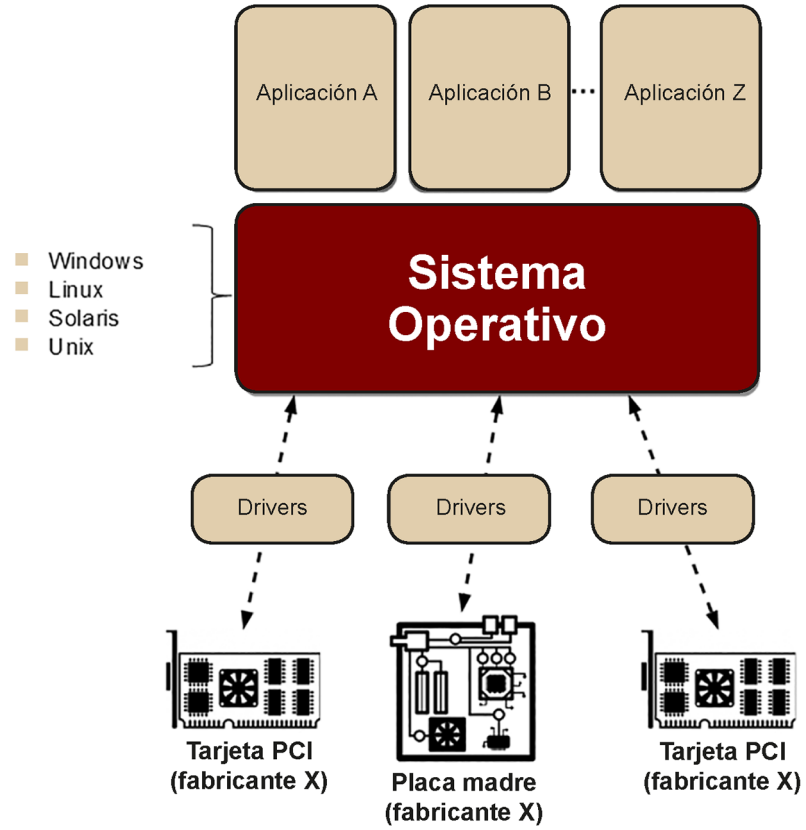
1.2. Comprendiendo el cambio de paradigma que las SDN introducen

Para comprender bien el cambio de paradigma que las SDN introducen vamos a ver un símil que ayudará a entenderlo. Compararemos el modelo arquitectural de un ordenador a nivel de *software* con el modelo de las SDN. Como veremos a continuación, guardan muchas similitudes.

1.2.1. ¿Cómo está organizado un PC internamente a nivel de *software*?

Aunque es ampliamente conocido cómo se estructura a alto nivel un ordenador por dentro vale la pena echarle un vistazo. La Figura 1 muestra dicha arquitectura.

Figura 1. Arquitectura de un ordenador a nivel de *software*.



En el nivel más bajo, tenemos los diferentes componentes *hardware*, los cuales pueden ser de distintos fabricantes (tarjetas PCI, placa madre, etc.).

No obstante, para que el sistema operativo pueda controlarlos es necesario instalar los *drivers* que solo cada fabricante de cada componente nos puede facilitar. Es decir que los *drivers* vienen a ser un enlace entre el *hardware* y el sistema operativo enmascarando la heterogeneidad de los fabricantes en los distintos dispositivos *hardware*. A este enlace le llamaremos ***southbound*** teniendo en cuenta que el sistema operativo hace de "ecuador" entre dos hemisferios.

El **sistema operativo**, gracias a los *drivers*, realiza una **abstracción de los recursos *hardware*** con respecto a las aplicaciones que se ejecutan en él. Es conocido que hay diferentes tipos de sistemas operativos: Windows, Linux, Unix, etc. e incluso de cada sistema operativo podemos encontrar diferentes versiones o distribuciones (especialmente en sistemas operativos como Linux entre los que podemos encontrar, OpenSuse, Debian, Ubuntu, Red Hat, etc.). La base de estas distribuciones es la misma (el kernel) pero con pequeñas variaciones en las que se ofrecen nuevas prestaciones y aplicaciones de serie que facilitan tareas al desarrollador de aplicaciones.

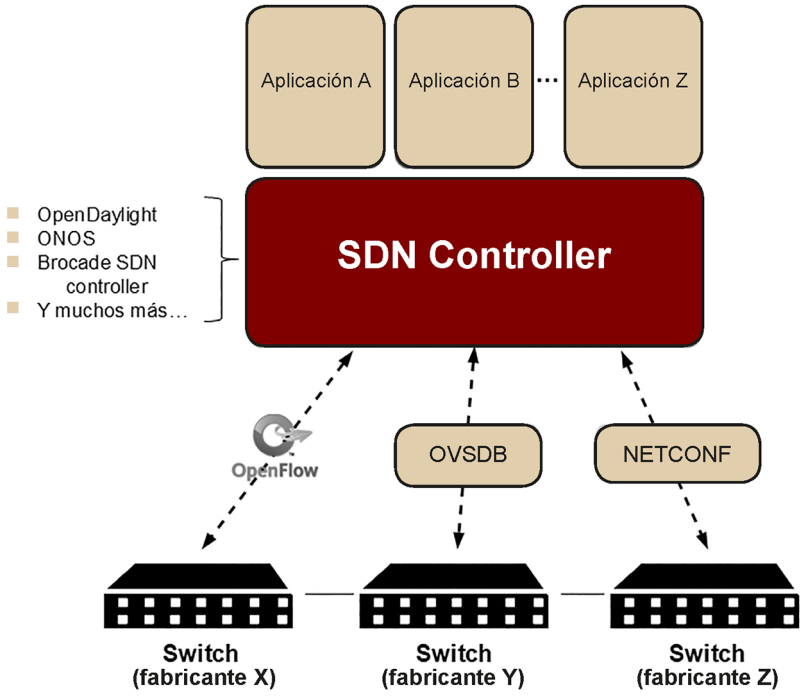
Finalmente, mediante APIs (*Application Programming Interface*) especialmente adaptadas al sistema operativo, se pueden desarrollar diferentes aplicaciones sin tener que preocuparse de las particularidades del *hardware* del ordenador. Estas APIs hacen de enlace entre la aplicación en sí y el sistema operativo

donde se ejecuta. A este enlace le llamaremos ***northbound*** ya que desde el punto de vista del emplazamiento del sistema operativo en esta arquitectura las aplicaciones están "sobre" él.

1.2.2. ¿Qué similitud guarda con el modelo arquitectural de las SDN?

La Figura 2 permite comprender enseguida la equivalencia entre ambos modelos.

Figura 2. Arquitectura equivalente en SDN.



En lugar de tarjetas PCI y placas madres lo que tenemos son toda una serie de ***switches* que físicamente están interconectados entre sí** con cableado de red pero sin ningún tipo de configuración cargada (a excepción del agente SDN que deben tener integrado y configurado para que pueda comunicarse con el correspondiente SDN Controller). La localización física de dichos *switches* puede ser muy dispar y pueden estar completamente separados del servidor donde se ejecuta el SDN Controller.

Así pues, tenemos desplegados varios elementos *hardware* de red que pueden ser todos de distinto fabricante.

En el lado ***southbound***, tenemos el enlace entre los *switches* y el SDN Controller via una conexión de tipo cliente-servidor que es implementado mediantes protocolos de M&C algunos específicamente dedicados para la implementación de las SDNs. Estos protocolos cumplen funciones asimilables a los *drivers* de la red. Entre estos protocolos podemos destacar uno en concreto: **OpenFlow** el cual se considera el estándar a soportar por casi cualquier fabricante.

M&C

Responde a las siglas de Monitorización y Control y se refiere a las funciones de configuración de un equipo y al envío de eventos en tiempo real asociados a su funcionamiento.

En paralelo han ido apareciendo más protocolos que también están siendo muy utilizados como OVSDB y NETCONF. Pero podemos encontrar muchos más.

En el SDN Controller encontramos un elemento que posee una visión holística de toda la red abstrayendo a las aplicaciones de los recursos de red. El **SDN Controller se le considera incluso como el sistema operativo de la red**. Y como sistema operativo nos encontramos con varias distribuciones, algunas open source (como Opendaylight o ONOS), y otras propietarias (como VMWare NSX o Cisco APIC). También podemos encontrar varias distribuciones basadas en estos controllers. Por ejemplo, Brocade, Avaya y Coriant son SDN Controllers comerciales que están basados en Opendaylight.

Por el lado ***northbound***, el SDN controller ofrece una serie de APIs a los desarrolladores de aplicaciones que facilitan su implementación. A diferencia de los protocolos de lado *southbound*, no existen APIs que se consideren estándares y cada aplicación deberá estar desarrollada basándose en el SDN Controller y las APIs que este ofrezca.

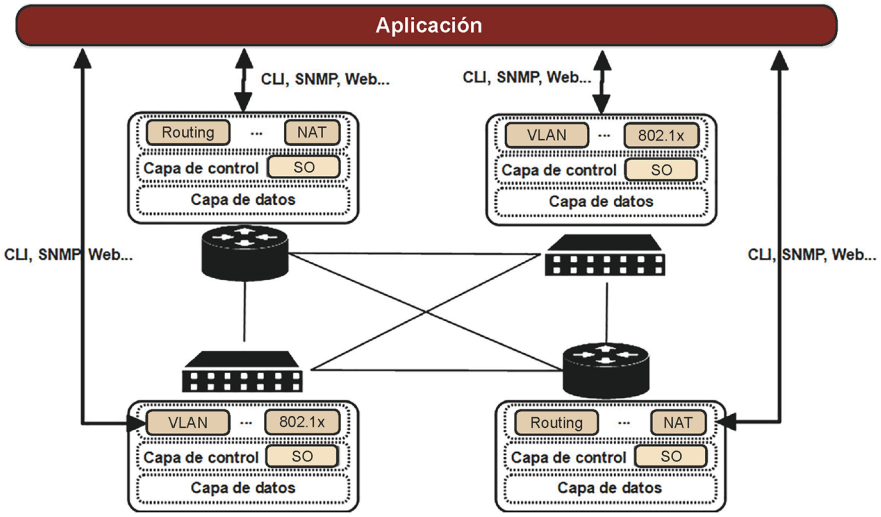
No obstante, los desarrolladores de aplicaciones no tendrán que lidiar con la complejidad de tener que configurar todos los elementos de red uno a uno ni con las particularidades de los interfaces de configuración (CLI particular, MIB SNMP particular) de cada fabricante.

1.2.3. El modelo de referencia de las SDN

Este símil ha ayudado a empezar a ver hacia dónde se encamina la arquitectura de las redes en el futuro y el papel que juega el SDN Controller pero vamos a centrarnos en el impacto que este nuevo paradigma va a tener en los elementos de la red que procesan paquetes y se encargan de dar "inteligencia" a la red: los *routers* y los *switches*.

El siguiente diagrama (Figura 3) describe a alto nivel la arquitectura de una red sencilla con un modelo previo a las SDN. El diagrama también describe la arquitectura funcional interna en capas de un *router* o un *switch* en una red que cumple el paradigma de inteligencia distribuida tradicional.

Figura 3. Modelo arquitectónico previo a las SDN.



Cada uno de los elementos que conforman la red tradicional (*routers* y *switches*) puede estructurarse internamente en varias capas que describimos a continuación:

- **Capa de Aplicación**: que implementa una funcionalidad o *feature* en concreto dentro del dispositivo. Por ejemplo, en el router, la funcionalidad de NAT (*Network Address Translation*) o IP *routing* estático o dinámico que en ocasiones implica el intercambio de mensajes entre *routers* adyacentes (protocolos RIP o OSPF), generados por esta capa.

- **Capa de Control**: viene a ser el sistema operativo del dispositivo y es quien toma decisiones de encaminamiento entre puertos y/o priorización de paquetes IP de acuerdo a las instrucciones provenientes desde las *features* en la capa de aplicación. Esta capa suele ser un *software* propietario que está totalmente asociado al *hardware* del equipo (suele ser un *software* propietario del fabricante).

- **Capa de Datos**: es simplemente la capa donde se transfieren los paquetes IP entrantes y salientes del *router* o las tramas Ethernet entre puertos entrantes o salientes en un *switch*. En esta capa se aplican las decisiones recibidas desde la capa de Control. Normalmente, esta información de transferencia entre puertos se almacena en una tabla específica optimizada para realizar el procesado paquete a paquete lo más rápido posible.

A nivel de arquitectura funcional **lo más distintivo de este modelo tradicional es que todas estas capas están integradas en el mismo dispositivo *hardware* y las decisiones de encaminamiento se deciden localmente**. No obstante, como se puede ver en la Figura 3, si se quiere implementar una aplicación global a nivel de red se deben configurar varios elementos de red simultáneamente, ya sea a través de un operador de red (ingeniero) o un *script* (proceso automatizado).

El operador (o *software* externo que implementa la aplicación) tiene que comunicarse individualmente con cada dispositivo y configurarlos uno a uno. Dicha conexión es directamente con la capa de aplicación de cada dispositivo usando los interfaces de configuración que el fabricante haya implementado (por ejemplo, puede ser vía línea de comandos mediante una conexión telnet o SSH o también vía el protocolo SNMP modificando el contenido de una MIB).

MIB

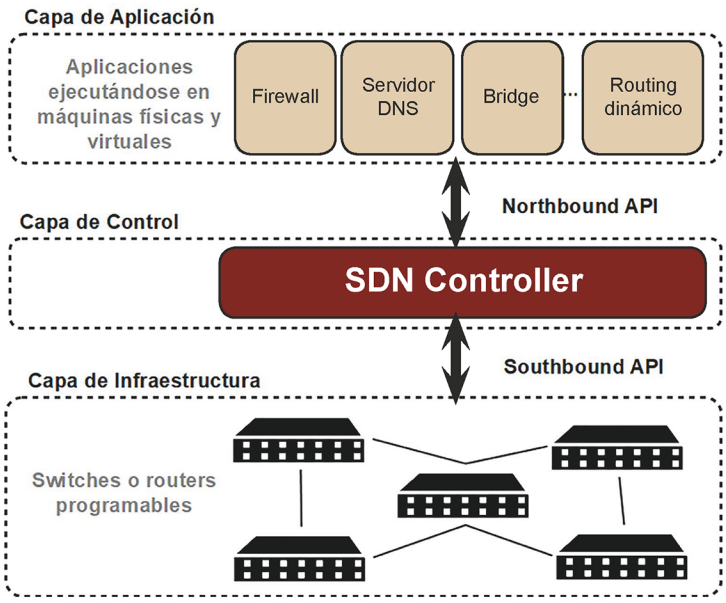
Responde a las siglas en inglés de *Management Information Base* y es un tipo de base de datos que contiene información jerárquica, estructurada en forma de árbol, de todos los parámetros gestionables en cada dispositivo gestionado de una red de comunicaciones.

Como ya hemos indicado en secciones previas la gestión de dichas redes, sobre todo en el caso de redes muy grandes donde hay muchos dispositivos, cuando hay que realizar reconfiguraciones o personalizaciones en la tipología para adaptarla a una aplicación en concreto implica también la reconfiguración de un gran número de equipos.

En definitiva, este modelo arquitectónico sufre de cierta **rigidez y falta de flexibilidad** para poder adaptar los recursos de la red a los requisitos de robustez, ancho de banda y QoS.

Lo que propone el nuevo paradigma de las SDN rompe completamente con esta arquitectura funcional y se muestra en la Figura 4.

Figura 4. Modelo arquitectural de las SDN.



Aunque este modelo ya se ha introducido en secciones anteriores veamos cual es la principal novedad que se introduce en este nuevo paradigma.

La principal novedad que introduce este nuevo paradigma es el **desacoplo físico entre la capa de control y la capa de datos o *forwarding*** que procesa y encamina los paquetes entre los puertos del *switch* o *router*.

Este desacoplo es a nivel físico. Es decir, que la capa de control y la capa de datos ya no se encuentran en el mismo dispositivo físico. Múltiples *switches*/*routers* se despliegan geográficamente mientras que el SDN Controller puede estar ubicado en un servidor de un *data center*.

Por un lado, tenemos un SDN Controller (solo uno, aunque puede estar redundado) que controla a un gran número de *switches* y/o *routers* habilitados para SDN (es decir, que implementen la API de *southbound*). Esto nos lleva a concluir que **un solo SDN Controller es dueño de un dominio entero al cual pertenecen diferentes *switches* y *routers* en la capa de infraestructura o de datos**.

1.2.4. Método que usa el SDN Controller para controlar los elementos de la capa de infraestructura

El SDN Controller se comunica directamente con los elementos de la capa de infraestructura (debidamente habilitados para SDN) a través de los protocolos de la API de ***southbound***. El SDN Controller es como si programara directamente la tabla CAM (*Content Addressable Memory*) de un *switch*. De hecho, en un contexto SDN el concepto de tabla CAM cambia ligeramente de nombre ya que se le llama **tabla de flujos** o ***flow table*** en inglés y a cada entrada que el SDN Controller programa se le llama **entrada de flujo** o ***flow entry***.

Southbound API

Hay múltiples protocolos para controlar remotamente a dispositivos habilitados para SDN como OpenFlow, NETCONF y OVSDB.

Estas tablas de flujos están formadas por campos de filtrado que abarcan tanto campos de la cabecera Ethernet como de la cabecera IP y TCP/UDP. Si un paquete hace *match* en una de estas entradas de flujos se le **aplica una acción**, la cual también es programada desde el SDN Controller.

Tabla CAM

Una tabla CAM le indica a un *switch* qué debe hacer con una trama que entra por un puerto X. Estas acciones sobre la trama pueden ser sacarlo por un puerto Y o entregarlo a la capa de control para realizar otro tipo de procesado más complejo.

Esta acción puede ser desde indicar el puerto de salida del *switch*, sin más, a enviar el paquete al SDN Controller (debidamente encapsulado en el protocolo *southbound* que se esté usando en dicho dispositivo). Normalmente cuando se envía un paquete al SDN Controller éste lo devuelve al mismo *switch* de donde ha partido una vez ha sido debidamente procesado por la aplicación. No obstante, dependiendo de la aplicación el propio SDN Controller puede ser el origen o destino de tráfico IP desde/hacia el *switch*.

> Por ejemplo, un SDN Controller puede hacer una comprobación de la interconexión de dos o más *switches* generando él mismo un paquete IP de tipo ICMP y configurando previamente entradas de flujos en los *switches* involucrados para que vuelva dicho ICMP al mismo SDN Controller por otro *switch* distinto. Así se comprueba que siga existiendo continuidad y que los cables sigan conectados entre ciertos *switches*.

Si el *switch* lo soporta, se pueden programar **otras acciones suplementarias antes de sacar el paquete por un puerto físico**. Estas acciones se pueden realizar en el mismo *switch* y pueden ir desde la modificación de campos de diversas cabeceras del paquete, hasta realizar tareas de limitación de flujo a una tasa de bit configurada.

Una red SDN que se considera bien implementada **ha de intentar configurar con antelación las máximas entradas de flujos posibles** que afecten al procesado de paquetes dentro de cada uno de los dispositivos gestionados **a la vez que se minimiza tener que reenviar paquetes al SDN Controller**, ya que puede tener efectos en la latencia.

En las siguientes secciones vamos a ver más detalladamente los elementos e interfaces más importantes de la arquitectura de referencia de las SDN.

1.3. El SDN Controller

El SDN Controller es el elemento que simboliza mejor el desacoplo entre el *software* y el *hardware* de la red y también, como se ha dicho anteriormente, entre la capa de control y la capa de datos o de *forwarding*.

El **SDN Controller es el principal cerebro y controlador de un entorno SDN y ofrece una visión centralizada de toda la red**, siendo así un punto estratégico de control dentro de la red y siendo responsable de comunicar información a:

- *Switches*, *routers* y otros elementos tras éstos. Para ello, el SDN Controller usa APIs o protocolos llamados de *Southbound* (como OpenFlow o NETCONF). Esta comunicación se realiza con cualquier *router* o *switch* independientemente del fabricante.

- Aplicaciones e incluso a la capa de aplicación y control de otros *routers* y *switches* que no estén gestionados por SDN. Para dicha comunicación se usan APIs u otros protocolos llamados de *Northbound*.

El SDN Controller se considera como un sistema operativo de red sobre el cual desarrollar múltiples aplicaciones. De hecho, sin una aplicación ejecutándose sobre un SDN Controller, éste, por sí solo, no hará ningún tipo de gestión ni configuración de la red. El SDN Controller carece de sentido si no hay aplicaciones ejecutándose.

Siguiendo la misma filosofía que un sistema operativo en un ordenador, un SDN Controller no es un *software* monolítico, sino que es una composición de múltiples programas, los cuales están estructurados en forma de módulos y *plugins*. Cada módulo puede realizar diferentes tareas de red que son a la vez utilizadas por las aplicaciones a modo de "servicios" que completan capacidades que las aplicaciones ofrecen.

SDN

Ved también la web https://www.sdxcentral.com como fuente de información sobre los conceptos que rodean SDN así como las últimas tendencias en este campo.

En el mercado existen múltiples SDN Controllers donde elegir. La elección de uno u otro para implementar un dominio SDN es algo que los administradores de red deben estudiar con detenimiento para garantizar estabilidad y soporte a largo plazo para futuras actualizaciones. Existen SDN Controllers tanto de código libre como propietarios. Otro criterio a tener en cuenta en la elección es el ecosistema de aplicaciones desarrolladas para cada SDN Controller en cuestión, así como la madurez de las *Northbound* APIs que éstos ofrecen.

OpenStack

Se trata un entorno para implementar un data center basado en *cloud computing*. Ofrece una plataforma de orquestación de máquinas virtuales y a la hora de interconectarlas entre sí se suele usar tecnología SDN y por lo tanto debe tener un enlace adaptado con un SDN Controller *(plugin*).

La elección del SDN Controller viene condicionada también por la finalidad de la red a gestionar. Se pueden encontrar SDN Controllers cuyas características, módulos y *plugins* incluidos están más focalizados para un caso de uso u otro.

Por ejemplo, no es recomendable utilizar un SDN Controller para un *data center* si éste ha sido adaptado para desplegarse en una red de un campus universitario. Del mismo modo, no tiene sentido elegir un SDN Controller para un campus que incluya un *plugin* específico para OpenStack.

¿Hay algún SDN Controller ampliamente adoptado?

Desde la aparición del primer SDN Controller (llamado NOX en 2009, el cual no era de código abierto), este tipo de controladores han ido evolucionando, apareciendo nuevos y también han ido desapareciendo a lo largo de los años (por obsolescencia y desuso por la comunidad de ingenieros). Por estos motivos es difícil hoy en día considerar un solo SDN Controller como dominante en el mercado, aunque sí que vale la pena mencionar a uno en concreto: **OpenDaylight**.

Opendaylight

Ved también la web https://www.opendaylight.org/

OpenDaylight (ODL) es un SDN Controller de código abierto impulsado hoy en día por la Linux Foundation, aunque originariamente, cuando la primera versión apareció en 2013, estaba liderado por una colaboración entre Cisco e IBM.

Distribuciones ODL y Linux

Se puede encontrar un paralelismo entre las distintas distribuciones de ODL como de Linux

Si destacamos éste es porque, a parte de ser un *controller* por sí mismo, se ha adoptado como base para desarrollar otros SDN Controllers comerciales a los que se les ha añadido funcionalidades extras propietarias.

Ejemplos de estos SDN Controllers comerciales basados en ODL son Brocade, Ericsson y Ciena entre otros.

También podemos destacar otro SDN Controller de código abierto que viene a retar la prevalencia de ODL. Es el ***Open Networking Operating System* (ONOS)**.

Este SDN Controller está focalizado en las necesidades de los proveedores de servicio (proveedores de conectividad WAN). Aunque al principio este SDN Controller no era de código abierto, se convirtió finalmente a esta modalidad

en el año 2014. Este producto está apoyado por numerosas organizaciones como AT&T, Cisco, Fujitsu, Ericsson, Ciena, Huawei, NTT, SK Telekom, NEC e Intel. Algunas de estas organizaciones también dan soporte a OpenDaylight.

1.4. *Southbound* APIs y desacoplo entre la capa de control y de datos

Las *Southbound* APIs son las que realmente hacen posible el desacoplo entre la capa de control y la capa de datos. Existen numerosas APIs en forma de protocolos que los SDN Controllers utilizan para programar a los *switches*. Pero entre todos estos protocolos hay uno que sin duda debe ser destacado: **OpenFlow.**

OpenFlow (OF) se considera **el estándar en cuanto a protocolos *southbound* porque fue el primero de todos**. Aunque hoy existen más *Southbound* APIs, éste es soportado por todos los SDN Controllers así como *switches/routers* con capacidades SDN.

De hecho, OF se creó incluso antes que los SDN Controllers en sí. OF fue creado por investigadores en redes de computadores en la universidad de Stanford en el año 2008 cuya primera versión (v1.0) fue publicada en diciembre de 2009. Ahora **el desarrollo de OF está liderado por la *Open Networking Foundation* (ONF)** que es una organización formada por usuarios que concentran su esfuerzo en avanzar en el desarrollo de estándares abiertos y la adopción de tecnologías SDN.

ONF

Ved también la web https://www.opennetworking.org/

OF es un protocolo que está específicamente diseñado para configurar tablas de flujos en un *switch*. Es decir, que básicamente configura entradas de flujos en diferentes tablas encadenadas (llamadas *pipelines* en inglés) y a cada entrada se le asocia una acción a realizar con el paquete. OF por ejemplo no es un protocolo para configurar direcciones IP en los interfaces ni para configurar políticas de QoS. Es decir, no es un protocolo de gestión. Intentando suplir las carencias de OF en cuanto configuración de dispositivos, ONF ha especificado un protocolo llamado **OF-Config**.

El siguiente protocolo *southbound*, cuyo uso está muy extendido, es el ***Open vSwitch Data Base* (OVSDB)**, el cual fue creado por Nicira. Originalmente, este protocolo era parte del Open vSwitch (OVS), el cual es un *switch* virtual de código abierto basado en Linux diseñado para hipervisores de máquinas virtuales. Ahora este protocolo ya no está enmarcado exclusivamente en el uso del OVS, sino que está implementado en otros switches de diversos fabricantes como Arista y Dell entre otros.

Nicira

Fue la empresa que creó NOX, el primer SDN Controller. Esta empresa fue después adquirida por Vmware.

En casos donde se despliega un OVS (tanto si está físicamente en un equipo como si está virtualizado), OpenFlow se sigue usando con este *switch* virtual para configurar las tablas de flujos que definen las reglas de traspaso y envío de

paquetes entre puertos. Sin embargo, OVSDB es usado para configurar el OVS en sí. Es decir, que se pueden crear, borrar y modificar puertos e interfaces, así como *bridges*. Esto es porque OVSDB se considera más un protocolo de gestión de un *switch* que un protocolo de control de un switch (como sí lo es OF).

Seguimos con el **Network Configuration Protocol** o **NETCONF**, que es un protocolo definido por el IETF para instalar, manipular y borrar la configuración de dispositivos de red. Este protocolo normalmente se ejecuta sobre un canal seguro usando encapsulados como SSH o TLS y conforma una serie de mensajes que contienen comandos (llamados RPCs o *Remote Procedure Call*) para cambiar el estado del dispositivo. Sin embargo, la información contenida en los mensajes de NETCONF no está definida por dicho protocolo, sino que está estructurada según un lenguaje de modelado de datos llamado YANG (*Yet Another Next Generation*). YANG no contiene datos en sí, sino que solo especifica cómo se estructuran. La información contenida en los comandos de NETCONF se codifica usando XML o JSON obedeciendo a la estructura que se ha definido previamente usando YANG.

NETCONF *vs* RESTCONF

Existe un protocolo llamado RESTCONF. Se puede decir que es lo mismo que NETCONF pero encapsulado en HTTP y usando los comandos que este protocolo proporciona (GET, PUT, POST, etc.). Es común utilizar RESTCONF como implementación de API *Northbound*.

YANG define cómo se estructuran los datos (estructuras en forma de árbol) y proporciona muchas características de modelado incluyendo un extenso catálogo de tipos de variables, configuración de datos y una variedad de reglas de semántica y sintaxis.

Las definiciones de datos están contenidas en módulos, lo que proporciona es un set muy potente de características para poder extender dichas definiciones y ser reutilizadas.

Estos modelos de datos basados en YANG pueden ser, por ejemplo, completamente definidos por el fabricante del *switch* o también pueden estar basados en modelos ya existentes que han sido definidos por organizaciones de estandarización (como el IETF o IEEE, que definen modelos genéricos para configurar los interfaces, rutas IP así como políticas de QoS, entre otros) o también por un grupo de trabajo llamado OpenConfig, formado por proveedores de servicio y operadores de telefonía que intentan suplir los huecos que las organizaciones de estandarización han dejado.

YANG y ASN.1

ASN.1 era un lenguaje de modelado de datos de propósito general usado para implementar la estructura de una MIB de SNMP. YANG ofrece un lenguaje mucho más entendible por el hecho de ser legible por un humano y permite definir muchas más cosas, incluso los mensajes que se intercambian entre dos elementos de red.

1.5. *Northbound* APIs y el desarrollo de aplicaciones

Como ya hemos comentado en anteriores secciones, el principal objetivo de las *Northbound* APIs (NBI) es la de proporcionar una capa de abstracción que permita a los desarrolladores de aplicaciones poder anclarse a la red y realizar cambios en ésta para acomodar las necesidades de la aplicación sin tener que entender exactamente qué equipos de red hay que modificar.

Aunque el cambio principal de paradigma que las SDN ofrecen se implementa gracias a las *Southbound* APIs, es muy importante recalcar el papel que tiene las *Northbound* APIs.

Las *Northbound* APIs son posiblemente las más críticas ya que son las que aportan el valor añadido a las SDN, vía las aplicaciones innovadoras que se desarrollan.

Actualmente los protocolos REST (*Representational State Transfer*) parecen ser los más utilizados para implementar las *Northbound* APIs y la mayoría de los SDN Controllers los implementan. Los mensajes de un protocolo implementado como REST contienen información en formato XML o JSON, los cuales pueden seguir un modelo basado en YANG.

Así pues, se puede decir que cada *Northbound* API da soporte a un tipo de aplicación en concreto y esa es posiblemente la razón por la que éstas APIs son el componente menos estandarizado en el contexto de las SDN.

La ONF ha abierto un grupo de trabajo específicamente para las *Northbound* APIs. Esta organización pretende desarrollar código y prototipos para evaluar si es viable o no desarrollar un estándar en este componente.

Consecuentemente existirán en el SDN Controller un amplio abanico de interfaces para controlar diferentes tipos de aplicaciones.

Nivel de abstracción que ofrecen las *Northbound* APIs

Es relativamente común que los SDN Controllers ofrezcan por defecto una serie de aplicaciones sencillas en forma de *plug-ins* que ayudan a desarrollar nuevas aplicaciones. Estos *plug-ins* se pueden usar vía las respectivas NBI, pero el abanico de funcionalidades que estas NBI ofrecen dependen mucho del nivel de abstracción con respecto a la capa de infraestructura.

De hecho, es posible encontrar *plug-ins* de muy bajo nivel, como por ejemplo un *plug-in* relacionado directamente con los mensajes del protocolo OpenFlow (es decir, que la aplicación tiene la oportunidad de directamente gestionar los mensajes de este protocolo). Pero en cambio también podemos encontrar NBIs de muy alto nivel, que abstraen totalmente cualquier información de topología y de la tecnología subyacente (por ejemplo, estos *plug-ins* pueden ofrecer una API que contiene comandos de configuración de políticas de QoS y rutas IP, donde las políticas se aplican en un equipo y las rutas en otro equipo distinto sin que el usuario sea consciente de ello).

Siguiendo esta última premisa, podemos afirmar que la comunidad que se encarga del desarrollo de nuevas NBI está siguiendo dos corrientes:

- **Interfaces relacionados con la tecnología subyacente**: estas NBI tienen un nivel bajo de abstracción y publican capacidades específicas que el sistema de red subyacente puede proporcionar y gracias a una combinación de ellas los usuarios pueden realizar aplicaciones de red. Para poder utilizar estas APIs el usuario debe tener conocimientos de configuración de equipos (configuración de flujos IPs directa en dispositivos) debido al bajo nivel de abstracción, ya que se deben utilizar estas NBIs para indicar cómo ofrecer un servicio de red.

  IETF ha hecho muchos esfuerzos por conseguir NBIs estándares de este tipo. IETF ha generado por ejemplo un protocolo llamado ALTO (*Application-Layer Traffic Optimization*) que proporciona información de red (proporcionando servicios de red como vistas de la arquitectura de red simplificadas, vistas de los recursos de la red incluyendo costes). También se pueden encontrar otros servicios en forma de NBI como VPNService, que permiten definir una red virtual aislada de uso muy típico en redes virtualizadas de *data centers* o sistemas de orquestación (por ejemplo, al crear dos máquinas virtuales y necesitar que estén interconectadas en la misma subred).

- **Interfaces agnósticos a la tecnología subyacente**: estas NBI tienen un nivel más alto de abstracción y suelen llamarse en inglés *intent NBI*. El usuario de estas APIs se focaliza en lo que el servicio necesita ofrecer específicamente sin tener que preocuparse de configuraciones de más bajo nivel, reduciendo mucho la dificultad de operación de la infraestructura de red. En estas APIs no hay que especificar cómo ofrecer el servicio sino que simplemente ofrecen el servicio en sí. Estas NBIs son mucho más complejas ya que llevan integradas capacidades de orquestación de otros *plug-ins* que buscan ese nivel tan alto de abstracción.

  Existe un ejemplo de este tipo de APIs que vale la pena resaltar. OpenStack necesita de un NBI específico para poder interconectar dos o más máquinas virtuales entre sí formando una red completamente virtual. Esta NBI se llama Neutron.

OpenStack

El OpenStack, desarrollado en colaboración entre la NASA y RackSpace, es actualmente el proyecto de *software* de código abierto para la gestión de plataformas *cloud* más activo hoy en día.

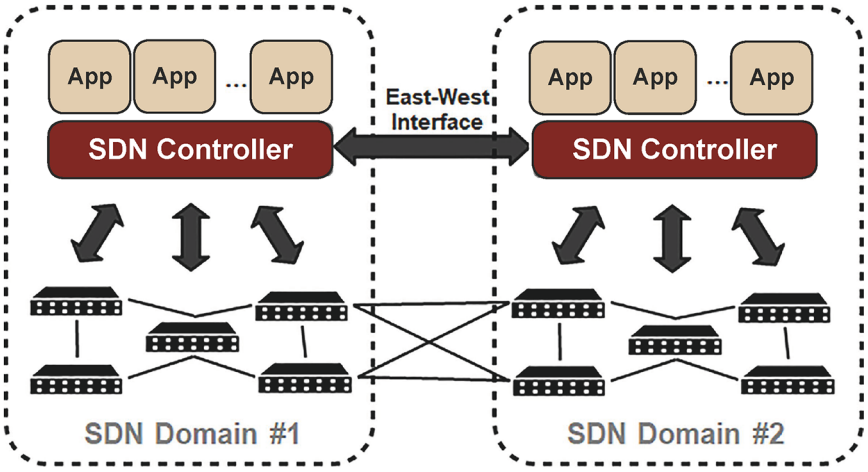
1.6. Interconexión entre SDN Controllers

Como ya hemos comentado en secciones anteriores desde el punto de vista de un SDN Controller existe una interfaz *Northbound* para desarrollar aplicaciones y una interfaz *Southbound* para controlar la capa de infraestructura. También hemos dicho que un SDN Controller gestiona un dominio entero que engloba a un número determinado de equipos de red en la capa de infraestructura.

Ahora bien, por razones obvias de escalabilidad (red multidominio), de capacidad de gestión, de separación de redes de dos operadores o simplemente por razones de privacidad, es posible que se utilicen más de un SDN Controller para controlar una red.

Dichos SDN Controllers deben poder comunicarse entre sí para coordinarse en el control de sus respectivos dominios y dicha comunicación es posible a través de una interfaz horizontal, llamada en inglés *East-West interface* o SDNi (ver Figura 5). Esta interfaz está gestionada por los propios *controllers*.

Figura 5. Interconexión entre dos SDN Controllers.



Los SDN Controllers pueden intercambiarse a través de esta interfaz SDNi información de condiciones y topología de su red, eventos, fallos de los propios *controllers*, requerimientos de QoS para ciertas aplicaciones, etc.

1.7. Ejemplo de uso de las SDN

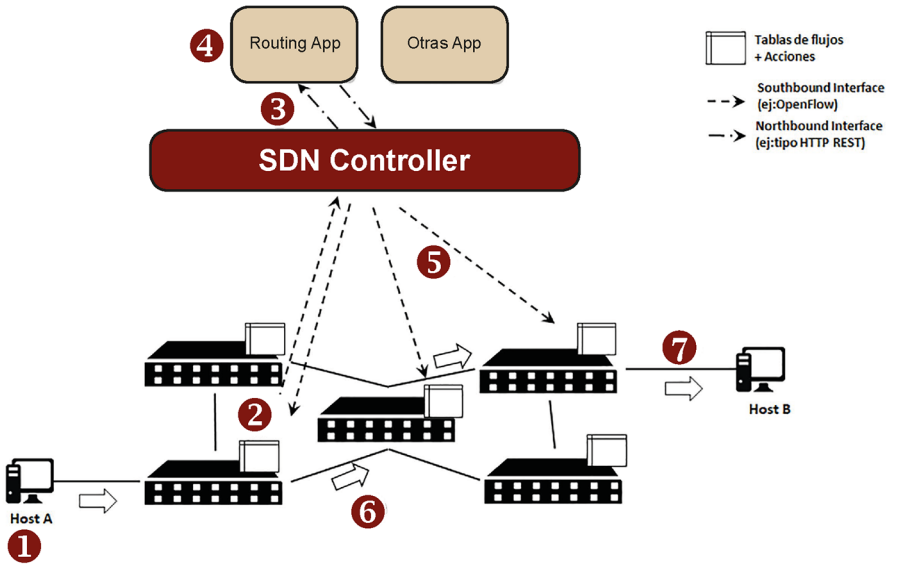
En esta sección mostraremos un ejemplo o caso de uso muy sencillo sobre cómo interactúa un SDN Controller con la capa de infraestructura y de aplicación.

En la Figura 6 tenemos una aplicación que realiza funciones de *routing* sencillo (*Routing App*) que interacciona con el SDN Controller vía una NB API (*Northbound* API) basada en HTTP REST. A través de esta API el *controller* recibe notificaciones provenientes de la capa de infraestructura, pero debidamente procesadas y orquestadas gracias al SDN Controller. Esta aplicación también tiene conocimiento de la topología de red ya que el SDN Controller la ha puesto a disposición de la aplicación.

El SDN Controller interacciona con la capa de infraestructura (*switches*) a través de una API *Southbound* (que podría ser OpenFlow, por ejemplo) y configura las tablas de flujos con filtros asociando acciones concretas para el caso de *match* o *miss* (es decir, cuando no hace *match*).

El caso de uso que proponemos es el de un *host* A que quiere enviar un mensaje instantáneo (paquete IP de pequeño tamaño) a otro *host* B, el cual se encuentra en un lugar remoto con respecto al *host* A.

Figura 6. Ejemplo de interacción entre la capa de infraestructura y el SDN Controller.



Veamos paso a paso lo que sucede a grandes rasgos:

1) El *host* A envía el paquete IP (no significa que A esté conectado directamente al primer *switch*, más bien queremos indicar por donde entra dicho paquete en el dominio que el SDN Controller gestiona).

2) El primer *switch* aplica los filtros de la tabla de flujo que tiene configurada en ese momento y detecta que el paquete recibido no hace *match* en ninguna entrada de flujo (*flow entry*) y por lo tanto no sabe por qué puerto sacar el paquete. Así que aplica la acción configurada por defecto que en este caso es enviar el paquete IP encapsulado en el canal de control de la SBI (por ejemplo, Openflow permite encapsular paquetes IP para hacerlos llegar al SDN Controller) y también solicitar que se tome una decisión de encaminamiento.

3) El SDN Controller realizar funciones de orquestación y sabe que dicho paquete debe ser enviado vía la NBI hacia la aplicación de *routing* específicamente (y no a otras aplicaciones que se pueden estar ejecutando también).

4) La aplicación recibe el paquete y según su información elabora la configuración de encaminamiento hasta el destino final (*host* B). Responde con el paquete IP y con las órdenes de configuración de rutas (según especifique la API del NBI).

5) El SDN Controller configura la tabla de flujos del primer *switch* enviando el paquete IP de vuelta para que salga por el puerto que toque, también configura las tablas de flujos con el *flow entry* más la acción correspondiente (en este caso la acción es indicar el puerto de salida) de todos los *switches* afectados que hay por el camino que se ha decidido.

6) Una vez configurados todos los *switches*, el paquete sigue su camino sin ninguna intervención más del SDN Controller con respecto a este flujo. Es decir, que, si hay que encaminar paquetes iguales al inicial, los *switches* usarán las tablas de flujos ya configuradas previamente (estas entradas en las tablas de flujos durarán hasta que se dispare un temporizador de inactividad, momento en el cual las *flow entry* se eliminan notificando al SDN Controller de ello).

7) El paquete llega a su destino.

2. *Network Function Virtualization* (NFV)

NFV responde a las siglas de *Network Function Virtualization* en inglés o **virtualización de funciones de red**. Pero antes de explicar este concepto sería útil especificar qué entendemos como funciones de red.

Las funciones de red se entienden como aplicaciones o servicios que cumplen con una función muy concreta y dan inteligencia y valor añadido al uso de una red.

Por ejemplo, estaríamos hablando de un cortafuegos, un servidor DNS, la funcionalidad de encaminamiento de un *router* (decisión de rutas IP) o también de balanceo de carga.

De hecho, se puede decir que estas funciones de red se enmarcarían en las ya mencionadas aplicaciones que se ejecutan en comunicación con un SDN Controller usando NBIs.

No obstante, entre estas funciones de red se contemplan arquitecturas más complejas, como por ejemplo un núcleo IMS (con todos los SIP proxies y elementos de procesamiento de señalización SIP y almacenamiento de perfiles que requiere), la parte de control y procesamiento de paquetes de una estación base de telefonía móvil o un EPC entero.

Hace unos años, era normal encontrar estas funciones en el mercado implementadas en plataformas *hardware* completamente propietarias, cuyo mantenimiento y configuración era costoso y, además, estas plataformas se mostraban como soluciones muy poco flexibles ante incrementos de demanda.

En el congreso *SDN and OpenFlow World Congress* del año 2012 en Darmstadt (Alemania) se propuso una solución: las NFV. Ese mismo año, tras dicho congreso, la ETSI creó un grupo de trabajo específico para estudiar su estandarización, el ETSI NFV *Industry Specification Group* (ISG). Este grupo se abrió a cualquier empresa miembro y no-miembro de la ETSI y desde entonces se ha avanzado mucho en dicha especificación.

NFV

Durante el *SDN and OpenFlow World Congress* de 2012 se publicó un *white paper* en el que se definió por primera vez el ecosistema de NFV (accesible en: https://portal.etsi.org/NFV/NFV_White_Paper.pdf).

En la web http://www.etsi.org/technologies-clusters/technologies/nfv se pueden seguir los avances del grupo de trabajo de la ETSI encargado de la especificación de NFV.

La **virtualización de las funciones de red** (o ***Network Function Virtualization*** en inglés) es una iniciativa que busca la implementación de servicios y aplicaciones de red en máquinas virtuales que se ejecuten en un *hardware* barato, fácilmente intercambiable y ampliamente disponible en el mercado (llamado ***hardware* de propósito general** o *commodity hardware* en inglés).

A cada una de estas aplicaciones o funciones de red monolíticas virtualizadas se les llama **VNF** o ***Virtual Network Function***. Varias VNFs (máquinas virtuales) pueden ser ejecutadas en el mismo *hardware* siendo monitorizadas y controladas por un hipervisor. En definitiva, se puede decir que es la aplicación del *cloud computing* para la provisión de servicios de telecomunicaciones.

¿Qué beneficios trae la introducción de NFV?

A continuación, vamos a describir algunas de las principales ventajas de la introducción del modelo de referencia de NFV:

- **Reducción de costes de implementación (CAPEX) y operación (OPEX):** el hecho de ejecutar *software* en un entorno de máquinas virtuales posibilita utilizar *hardware* de propósito general que es mucho más barato que el *hardware* propietario. Además, en un entorno así, el mantenimiento del entorno para una correcta provisión de servicios requiere menos personal y agiliza los procedimientos de operación.

- **Rapidez de adaptación a variaciones de demanda:** la virtualización permite poder iniciar casi instantáneamente la ejecución de nuevas instancias de un servicio adaptándose a los incrementos o decrementos (esporádicos o permanentes) del número de usuarios. Además, permite de manera muy fácil la incorporación e integración de nuevas unidades de *hardware*, aumentando potencialmente la capacidad de procesado ante futuros incrementos de demanda.

- **Reducción del tiempo de implementación de un servicio (*Time-to-market*):** si un proveedor de servicio quiere implementar un servicio nuevo, ya sea fruto de funciones de red aisladas (es decir, VNF que implementa todo un servicio) como fruto de encadenamiento de servicios (*service chaining* en inglés), el hecho de poder ejecutar *software* de manera virtualizada permite la composición y puesta en marcha de un nuevo servicio casi al instante.

- **Aislamiento total entre diferentes implementaciones del mismo servicio:** es lo que se llama en inglés *multi-tenancy*. Es decir, que se pueden ofrecer servicios totalmente adaptados y dimensionados a un grupo de usuarios en concreto. Se posibilita además que la ejecución de diferentes

instancias de funciones de red no se afecte mutuamente gracias a mecanismos de aislamiento de *software* entre dominios administrativos (los recursos de computación, almacenamiento y de red están totalmente compartimentados).

- **Reducción del consumo de energía eléctrica:** el desacoplo que la virtualización habilita entre la plataforma *hardware* y el *software* de función de red permite una utilización óptima de los recursos de computación (se ejecutan varias máquinas virtuales en un solo *hardware* exprimiendo al máximo sus capacidades sin desperdiciar recursos de CPUs y memoria sobrantes).

  Por ejemplo, si durante la noche la demanda de un servicio cae considerablemente, se puede concentrar la ejecución de *software* en menos servidores pudiendo apagar el resto de las plataformas o poniéndolos en modo de ahorro de energía.

2.1. Arquitectura de referencia de NFV

La ETSI, a través de su grupo de trabajo específico para NFV, ha definido una arquitectura de referencia para entender bien qué funcionalidades forman este ecosistema y cómo se relacionan/comunican entre sí.

Los criterios clave en los que se basó este grupo de trabajo para elaborar esta arquitectura de referencia son:

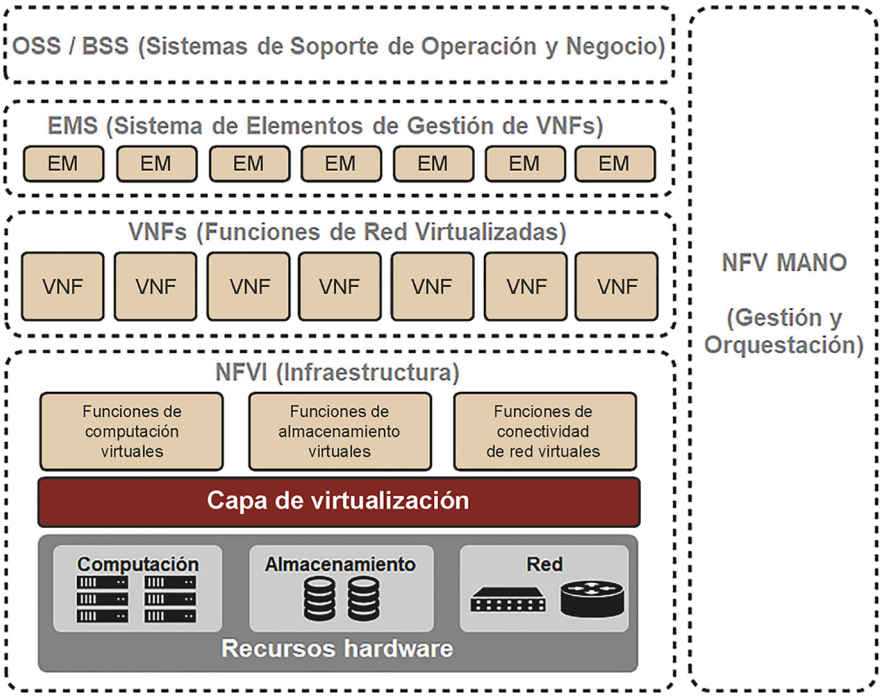
- **Desacoplo:** completa separación del *software* con respecto al *hardware*.

- **Flexibilidad:** despliegue de funciones de red automatizadas y escalables.

- **Operatividad dinámica:** control de los parámetros operacionales de las funciones de red vía un control y monitorización granular del estado de la red.

La Figura 7 muestra dicha arquitectura.

Figura 7. Modelo arquitectural de NFV.



A continuación, describiremos cada uno de los elementos principales de esta arquitectura de referencia.

2.1.1. VNF (*Virtual Network Function*)

El VNF se considera el bloque básico del ecosistema NFV. Es un elemento de red que se ejecuta en un entorno virtualizado (máquina virtual). Este *software* puede ser desde un *router* (el cual se podría llamar "*Router* VNF") a una estación base entera de telefonía móvil.

Aunque una VNF se concibe como la implementación de una funcionalidad monolítica en forma de *software* virtualizado, varias subfunciones de una funcionalidad de red pueden ser también implementadas como VNFs independientes.

> Por ejemplo, un núcleo IMS virtualizado puede estar compuesto por varias VNFs cada una de ellas implementando un elemento del núcleo: P-CSCF VNF, I-CSCF VNF, S-CSCF VNF y HSS VNF.

Hay que tener en cuenta que cada VNF es como una máquina virtual que se crea/instala, se ejecuta y finalmente se destruye cuando ya no se necesita. En todo este ciclo de vida de la VNF el sistema de gestión y orquestación (NFV MANO) se encarga de gestionarla y controlarla.

¿Quién implementa las VNFs?

Diferentes proveedores pueden implementar y comercializar VNFs para que luego un proveedor de servicio las utilice y/o combine para ofrecer un servicio a los usuarios.

Como veremos en posteriores secciones NFV MANO se encargará de orquestar el uso de recursos que los diferentes VNFs consumen con el objetivo de realizar ***service chaining*** o **encadenamiento de servicios**. Es decir, que la provisión de un servicio puede ser el resultado de la combinación y/o concatenación de varias VNFs cada una de las cuales implementa una función de red en concreto.

Hay proveedores de VNFs que implementan VNFs que por sí solas y sin necesidad de realizar *service chaining* ya ofrecen una función de red completa y suficiente para ofrecer un servicio.

Se llama *VNF set* al grupo de VNFs desplegadas por un proveedor de servicio, que no tienen ninguna interrelación entre sí (no hay secuencia de encadenamiento fija entre ellas). En un *VNF set,* varias VNFs pueden trabajar de manera colaborativa para ofrecer un servicio, pero su implementación es completamente independiente. Sin embargo, cuando en un servicio se requiere el procesamiento de datos en una secuencia específica entonces es cuando se llama *service chaining*.

2.1.2. NFVI (*Network Function Virtualization Infrastructure*)

El NFVI es el entorno en el que se ejecuta una VNF. Esto incluye recursos físicos, recursos virtuales y una capa de virtualización entre ambos, de ahí su nombre: Infraestructura. A continuación, describimos cada uno de dichos recursos:

- **Recursos físicos:** como recursos físicos entendemos todo aquello que es *hardware* y que lleva consigo capacidades brutas de procesado de paquetes, que luego la capa de virtualización se encargará de repartir entre las distintas VNFs. Se identifican tres tipos de recursos físicos: recursos de **computación** (servidores de propósito general), de **almacenamiento** (servidores de almacenamiento) y de **red** (*switches* y *routers* físicos) junto con todo el cableado que los interconecta entre sí.

Recursos físicos computacionales

Los recursos computacionales (CPUs y memoria) pueden venir en forma de apilamiento de servidores siguiendo técnicas de *cluster-computing*.

- **Recursos virtuales:** los recursos físicos son abstraídos en forma de recursos virtuales que son los que las VNFs realmente usan. De la misma manera, se identifican tres tipos de recursos virtuales: de **computación**, de **almacenamiento** y de **red**.

Recursos físicos de almacenamiento

Los recursos de almacenamiento pueden venir en forma de dispositivos NAS (*Network Attached Storage*) o dispositivos conectados a una SAN (*Storage Area Network*).

- **Capa de virtualización:** esta capa es la responsable de desacoplar el *software* del *hardware* haciendo posible que el *software* pueda progresar independientemente del *hardware* y viceversa. Existen dos metodologías de virtualización: basado en **hipervisores** (*hypervisor* en inglés) o basado en **contenedores** (*containers* en inglés). En ambos casos, los recursos físicos virtualizados son los computacionales y los de almacenaje mientras que, en el caso de los recursos de red, este elemento es el SDN controller que ya conocemos.

Un ejemplo muy conocido y consolidado de virtualización basado en contenedores es el propuesto por OpenStack, llamado Docker.

Diferencia entre *hypervisor* y *container*

Un hipervisor ejecuta máquinas virtuales que tienen su propio sistema operativo sobre un *hardware* que es virtualizado. Un *container* es mucho más ligero (ocupa mucho menos que una máquina virtual) y se ejecuta siempre sobre un sistema operativo a nivel de *host*.

2.1.3. EMS (*Element Management System*)

Antes hemos hablado de las VNFs como el *software* que implementa una funcionalidad de red en concreto en la provisión de un servicio y que este *software* se ejecuta en un entorno virtualizado.

FCAPS

Responde a las siglas de *Fault* (averia), *Configuration* (configuracion), *Accounting* (trazas), *Performance* (rendimiento) y *Security* (seguridad).

Pero estas VNFs deben estar configuradas y gestionadas para que cumplan con su función. Éste es el papel del sistema EMS. Este sistema alberga el *software* (también creado en tiempo real cuando una VNF se crea), que se encarga de gestionar a **nivel funcional** (cubriendo FCAPS) cada una de las VNFs Security que se ejecutan en el ecosistema NFV. La interfaz de gestión puede ser estándar o propietaria de la VNF en particular (y por lo tanto proporcionada por el mismo fabricante de la VNF).

Puede haber un EMS por VNF o también un EMS que pueda gestionar múltiples VNFs. El EMS en sí puede ser un VNF también. Sin embargo, el modelo de referencia de NFV a muy largo plazo contempla que el EMS desaparezca, pasando el testigo de la total gestión de las VNFs a manos de un elemento orquestador de VNFs dentro del NFV MANO: el VNFM o VNF Manager, que más tarde describiremos.

En un paso intermedio a esa migración final, los EMS pueden ser usados directamente por el subsistema OSS para configurar algunas VNFs o incluso para gestionar equipos físicos (llamados PNFs o *Physical Network Functions*). La ETSI contempla también que, de manera provisional, una versión reducida del EMS pueda asistir al VNFM a gestionar funcionalmente una VNF en concreto (sobretodo si se trata de una interfaz de gestión propietario entre EMS y VNF).

2.1.4. OSS/BSS (*Operation Support System/Business Support System*)

OSS/BSS incluyen una colección de sistemas y aplicaciones que un proveedor de servicios utiliza para operar su negocio.

Por un lado, OSS (*Operation Support System*, Sistema de Soporte de Operaciones) es el elemento que se encarga de la gestión de todo el sistema a nivel de red. El OSS es el sistema que utilizaría un operador de telecomunicaciones para gestionar toda la infraestructura de red que administra. Por ejemplo, este elemento se encargaría de los procesos de suministro de equipos, así como su configuración y también del registro de averías e incidencias.

Por otro lado, BSS (*Business Support System*) o Sistema de Soporte de Negocio se encarga de la gestión que tiene que ver con los clientes. Es decir, el sistema que se encarga de registrar las peticiones de servicio provenientes de los clientes, así como de facturar los servicios consumidos por estos.

En la arquitectura NFV, el subsistema OSS/BSS puede estar integrado con NFV MANO (que se verá con más detalle más adelante) mediante interfaces estandarizados. Esta integración con NFV no está exenta de retos que afectan a ambos subsistemas y en especial al de OSS. En este subsistema siempre se asumía que la infraestructura de red era más bien estática y su modificación/ampliación debía ser debidamente planificada. En definitiva, el OSS no había estado pensado para adaptarse rápidamente a cambios en la tipología y a la inclusión de nuevos servicios.

Para aprovechar la flexibilidad que ofrece NFV, el subsistema OSS necesita ser automatizado y basado en un **concepto de catálogos** (el cual veremos qué significa en la siguiente sección). Además, el OSS también necesita tener la capacidad de actualizar en tiempo real su conocimiento de los recursos disponibles. Consecuentemente, éste necesita también tener implementados mecanismos muy rápidos y automatizados para aplicar nuevas configuraciones en los equipos teniendo en cuenta esta capacidad de actualización en tiempo real.

2.1.5. NFV MANO (*MANagement & Orchestration*)

MANO responde a las siglas en inglés de *MANagement & Orchestration*. Esta parte del modelo de referencia introducido por el ETSI sobre las NFV es muy importante, pues se le considera el corazón y cerebro de toda la arquitectura.

Como se puede ver en la siguiente figura, la arquitectura de NFV MANO se subdivide en tres sub-bloques que posteriormente describiremos con detalle:

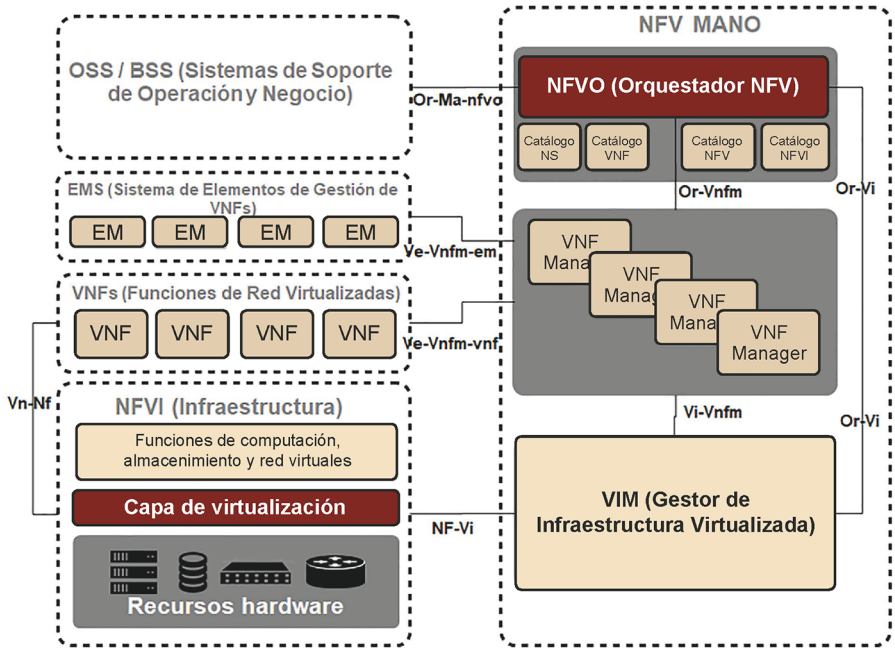
- VIM (*Virtualized Infrastructure Manager*) o Gestor de Infraestructura Virtualizada.

- VNFM (*Virtual Network Function Manager*) o Gestor de VNFs.

- NFVO *(NFV Orchestrator*) o Orquestador de NFV.

Cada uno de estos sub-bloques interacciona, a nivel de gestión, con su homólogo de la parte de la arquitectura que ya hemos descrito anteriormente.

Figura 8. Modelo arquitectural de NFV MANO.



VIM (*Virtualized Infrastructure Manager*)

Este elemento es responsable de gestionar todos los recursos (computación, almacenamiento y red) dentro del dominio administrativo de infraestructura de un operador o también llamado **dominio NFVI**.

Dominio NFVI

En una sola arquitectura NFV puede haber más de un dominio NFVI y cada uno de ellos gestionado por su correspondiente VIM.

Dicho de otra manera, el VIM se encarga de crear, mantener y destruir máquinas virtuales con respecto a los recursos físicos en dicho dominio NFVI. Cada una de estas máquinas virtuales tendrá asociados una serie de recursos *hardware* a través de la capa de virtualización. Así, el VIM mantiene un inventario de todas las máquinas virtuales asociadas a cada recurso físico.

El VIM se encarga también de mantener un inventario de todos los dispositivos *hardware*,así como de monitorizar y controlar el rendimiento y los fallos de dichos recursos *hardware*, *software* y funciones de recursos virtualizadas.

Este sub-bloque implementa su propia NB API para así exponer los recursos disponibles tanto físicos como virtuales a otros elementos de gestión dentro de la arquitectura NFV y en especial al NFV Orchestrator.

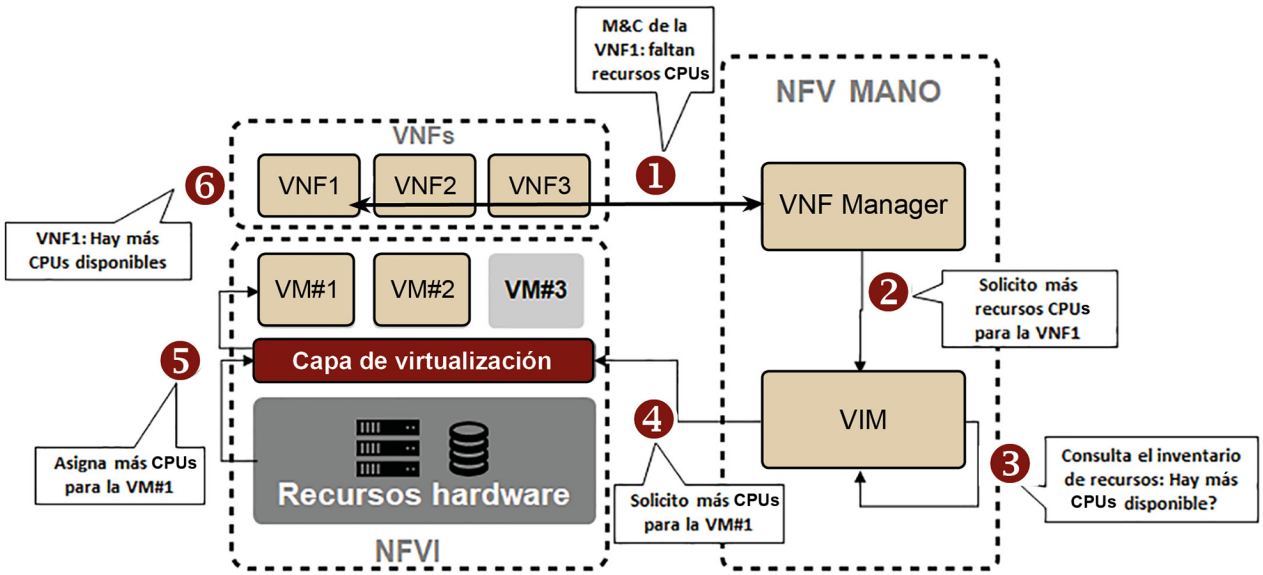
VNFMs (*Virtual Network Function Managers*)

Los VNFMs gestionan todas y cada una de las VNFs que se ejecutan a lo largo de todo su ciclo de vida. Es decir, crean, mantienen y destruyen cada una de las instancias de VNF. Estas VNF necesitan ser instaladas en máquinas virtuales, las cuales son creadas y gestionadas por el VIM, como ya hemos visto.

Relacionado con la gestión, una VNFM se encarga de cubrir las necesidades de gestión y monitorización a nivel de configuración, trazas, rendimiento y seguridad de cada VNF. Incluso se encargan de solicitar al VIM el incremento o liberación de recursos (computacionales o de almacenamiento) según necesite la VNF que gestiona. Se puede ver un ejemplo en la Figura 9.

Cada VNF puede disponer de su propio VNFM o un solo VNFM puede gestionar múltiples VNFs.

Figura 9. Ejemplo de interacción entre elementos de NFV para solicitar más recursos CPUs.



NFVO (*NFV Orchestrator*)

Este elemento es la piedra angular de NFV ya que aglutina toda la inteligencia de la arquitectura de referencia y porque su función es la de crear servicios *end-to-end* utilizando todos los recursos disponibles (NFVI) y las funciones de red virtualizadas (VNFs).

Se puede decir que el NFVO realiza dos tipos de orquestación:

- **Orquestación de recursos:** gestiona y tiene una visión global de todos los recursos que el VIM pública vía su *Northbound* API dedicada. El NFVO puede gestionar varios VIMs de distintos dominios NFVI. Así, se encarga de coordinar, autorizar, liberar y asignar dichos recursos.

- **Orquestación de servicios:** esta orquestación se encarga de realizar la concatenación de funciones de red (es decir, concatenación de VNFs) para ofrecer un servicio *end-to-end*. Estas VNFs serán gestionadas por sus respectivas VNFMs. Éstas VNFMs son instanciadas por este orquestador las cuales a su vez iniciarán las VNFs. El NFVO tiene conocimiento de la topología de red para cada instancia de servicio *end-to-end* (es decir, cómo las VNFs

están interconectadas entre sí para un servicio en concreto). Dicha topología se llama *VNF-Forwarding-Graph (*VNF-FG*)* o también *service chaining*.

Repositorios

Existen cuatro tipos de repositorios:

- **Catálogo de VNFs:** este catálogo alberga todos los descriptores de VNFs que son usables (VNFD: *VNF Descriptor*). Un descriptor de VNFs es como una plantilla que describe una VNF en términos de despliegue y requerimientos de operatividad. De manera genérica se pueden identificar tres parámetros que identifican un VNFD:
  - **VDU** (*Virtual Deployment Unit*): donde se especifica el nombre de la imagen donde se ubica la VNF así como los requisitos de CPUs, memoria y almacenamiento recomendados para ejecutar dicha VNF en una VM.
  - **CP** (*Connection Points*): donde se especifica el número de interfaces de red (virtuales) que la VM debe habilitar para la VNF. Cada CP debe ir asociada a una VDU y a un enlace virtual o *virtual link*.
  - **VL** (*Virtual Link*): se puede indicar opcionalmente el identificador del nodo virtual al cual conectar cada CP que incluya la VNF. La interconexión a este VL puede ser sustituida por la aplicación de un descriptor de servicio de red para interconectar varias VNFs.

Quien usa este descriptor es principalmente el VNFM durante el proceso de la instanciación de la VNF y la gestión de su ciclo de vida. También es usado por el NFVO para gestionar y orquestar los servicios de red, así como los recursos virtualizados en el NFVI (vía el VIM, como ya sabemos).

- **Catálogo de Servicios de Red:** alberga otro tipo de plantilla más relacionada con los servicios de red que se necesitan para interconectar VNFs. Los descriptores (NSD o *Network Service Descriptor*) en este catálogo describen la conectividad entre VNFs a través de enlaces virtuales. De esta manera pueden de un solo golpe interconectar una serie de VNFs para proveer un servicio en concreto.

- **Instancias NFV:** es una lista actualizada y detallada de todas las instancias creadas y activas tanto de VNFs como de Servicios de Red.

- **Recursos NFVI:** es un repositorio sobre los recursos NFVI utilizados con el propósito de ejecutar servicios VNF.

2.2. Puntos de referencia en NFV

Como se puede ver en la Figura 8, hay una serie de interfaces o puntos de referencia que comunican entre sí los bloques de NFV-MANO con los elementos restantes de la arquitectura de referencia de NFV.

Durante la explicación en secciones anteriores ya hemos dejado entrever la funcionalidad de cada uno de estos interfaces, pero a continuación daremos un resumen de las funcionalidades de cada uno de ellos:

- **Os-Ma-nfvo:** define la comunicación entre OSS/BSS y el NFVO. Se usa para la descripción de servicio e indicar los grupos de VNFs necesarios para dichos servicios. También se usa para gestionar el ciclo de vida tanto del propio servicio como de las VNFs que participan en el servicio (cuando crearlas y cuando acabarlas) y definir quién está autorizado a acceder a la instancia del servicio de red. Finalmente, a través de esta interfaz se envían eventos hacia el OSS/BSS sobre el rendimiento y el uso de dichas instancias de servicios.

- **Ve-Vnfm-vnf:** esta interfaz define la comunicación entre el VNFM y el VNF y es usada por el VNFM para gestionar el ciclo de vida de la VNF y para intercambiar información de su estado (eventos) y de su configuración. Por ejemplo, el VNFM puede instanciar, actualizar, escalar y eliminar las máquinas virtuales (VM) donde se ejecutan las VNFs.

- **Ve-Vnfm-em:** esta interfaz fue definida junto a la interfaz Ve-Vnfm-vnf pero ahora se ha separado de ésta para especificar la comunicación entre el VNFM y los bloques funcionales del EM (*Element Management*). Esta interfaz soporta la gestión del ciclo de vida de la VNF, pero solo se usa si el EM es integrable con el concepto de virtualización que NFV propone. A través del EM el VNFM también puede instanciar, actualizar, escalar y eliminar las máquinas virtuales (VM) donde se ejecutan las VNFs.

- **Nf-Vi:** esta interfaz define el intercambio de información entre el VIM y los bloques del NFVI. El VIM la usa para asignar, gestionar y controlar los recursos de NFVI. El VIM puede iniciar, actualizar, migrar y eliminar máquinas virtuales (VMs). Éste también puede crear, configurar y quitar conexiones entre máquinas virtuales. Finalmente, el VIM recibe desde el NFVI eventos, registros de usos e información de configuración.

- **Or-Vnfm:** esta interfaz define la comunicación entre el NFVO y el VNFM y se usa para instanciar VNFs y otros tipos de información relacionado con el ciclo de vida de estas (retransmite eventos provenientes de las VNFs).

- **Or-Vi:** esta interfaz es usada por el NFVO para tener una vía de comunicación con el VIM e influir en la gestión de los recursos de infraestructura (reserva, actualización y liberación de recursos para máquinas virtuales o

la instalación y desinstalación de VNFs). Finalmente se usa para reenviar eventos e información de configuración desde el NFVI al NFVO.

- **Vi-Vnfm:** define las directrices para el intercambio de información entre el VIM y el VNFM como por ejemplo la solicitud y actualización de recursos de la NFVI para las máquinas virtuales que ejecutan una VNF. También se usa para recibir eventos y medidas de recursos NFVI usados por una VNF.

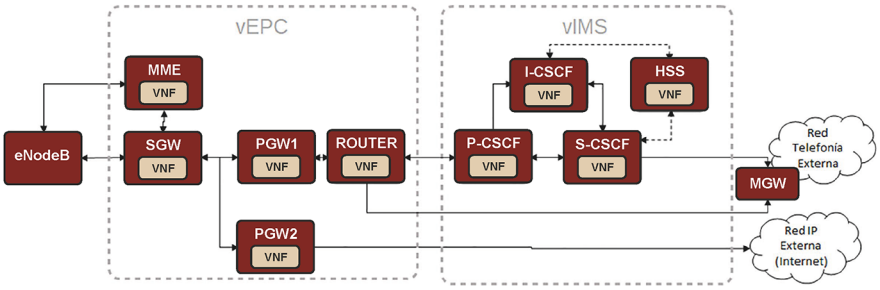
- **Vn-Nf:** esta es la única interfaz que no tiene una funcionalidad relacionada con la gestión. Este punto de referencia comunica necesidades de recursos y portabilidad de la VNF al bloque de infraestructura.

Hay que tener claro que la ETSI no ha definido ninguna recomendación de protocolo para implementar estos interfaces. Solo ha definido sus funcionalidades.

2.3. Ejemplo de *service chaining*: implementación de un EPC + un núcleo IMS

Una vez vista la arquitectura de referencia de NFV, es hora de exponer un ejemplo que enlace con el contenido del módulo anterior. Es por ello por lo que describiremos a alto nivel cómo se implementaría en un entorno virtualizado como el expuesto en este modelo de referencia un vEPC (*Evolved Packet Core* virtualizado) más un vIMS (núcleo IMS virtualizado).

Figura 10. Arquitectura a implementar en NFV: EPC virtualizado y núcleo IMS virtualizado.



En la Figura 10 vemos la arquitectura del servicio que se busca implementar:

- **EPC virtualizado:** compuesto por un MME que autentica los terminales de una eNB física que se encuentra fuera del ámbito administrativo del entorno NFV. Esta eNB también tiene una SGW asociada. Finalmente, este vEPC contiene dos PGW: una para el tráfico IMS y otra para la conexión a Internet.

- **Núcleo IMS virtualizado:** compuesto por un P-CSCF, un I-CSCF, un S-CSCF y un HSS. Para sacar las llamadas de voz a redes heredadas se cuenta con una MGW que tiene dos puntos de conexión físicos por separado:

uno para la señalización IMS y otra para el tráfico RTP de voz que llega directamente de la PGW2.

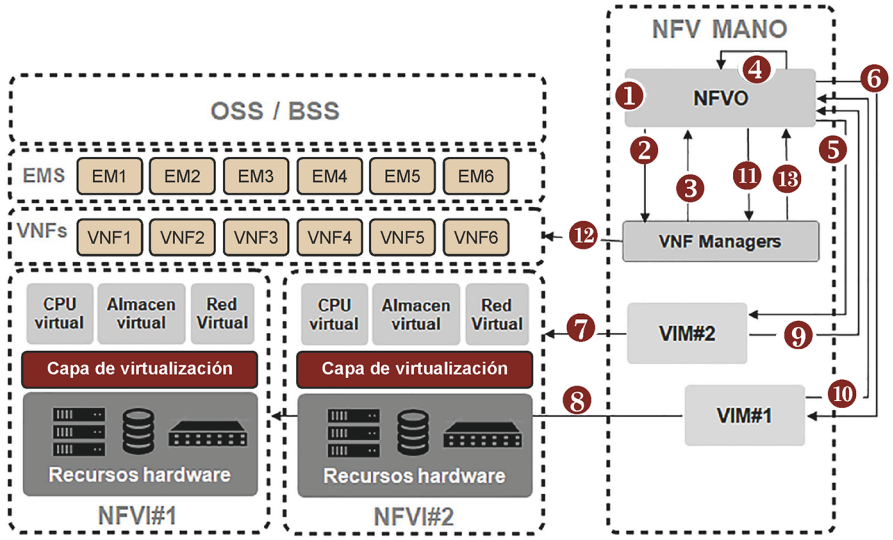
Para hacer el ejemplo más completo, el vEPC y el vIMS se implementan en dos dominios NFVI distintos, el segundo del cual es el que tiene conectividad con las redes externas: de telefonía vía la MGW y a Internet.

El proveedor de VNFs ha implementado una VNF para cada elemento del vEPC. Cada uno de los elementos funcionales que conforman el vEPC se implementa como una VNF. Así pues, tenemos en el catálogo de VNFs (en el NFVO) cada una de las de las VNFs que conforman un EPC. Lo mismo aplica a los elementos del núcleo IMS.

Cada VNF dispone de un descriptor en el que se indican requerimientos aproximados de CPUs, memoria, almacenamiento y red. En este descriptor aparecen también el número de puntos de conexión (*connection points* en inglés) de la VNF, es decir, cuantos interfaces de red externos se necesitan como mínimo para interconectarse con otras VNFs.

En el caso del catálogo de servicio de red se dispone de plantillas para establecer los enlaces virtuales (o *virtual links* en inglés) donde se detalla la interconexión a nivel de red entre VNFs para conformar un servicio.

Figura 11. Diagrama paso a paso de despliegue de VNFs para vEPC y vIMS.



A continuación, describimos paso a paso cómo se despliegan las VNFs para conformar tanto el vEPC como el vIMS:

1) El NFVO tiene una visión completa de la topología de las VNFs que conforman tanto el vEPC como del vIMS.

2) El NFVO accede al catálogo de VNFs, selecciona e instancia las VNFs de MME, SGW y dos PGWs para el vEPC, una VNF de *router* de interconexión

para la PGW1 y una VNF HSS, P-CSCF, I-CSCF y S-CSCF. Una vez seleccionadas se comunican a los VNFMs correspondientes (uno por VNF).

3) El conjunto de VNFMs determina el número de máquinas virtuales (VMs) necesarias para ejecutar todas las VNFs así como los recursos mínimos necesarios de cada una de estas VMs. Esta información se transmite de vuelta al NFVO.

4) Debido a que el NFVO mantiene información detallada y actualizada sobre los recursos *hardware* (en comunicación con ambos VIMs vía la NBI Or-Vi), valida si hay suficientes recursos disponibles para todas las máquinas virtuales (VMs) a ejecutar. El NFVO necesita instanciar una solicitud para disponer de dichas máquinas virtuales creadas.

5) El NFVO envia al VIM#2 la solicitud para crear las máquinas virtuales instanciadas para el vIMS y reservar los recursos necesarios para dichas VMs.

6) El NFVO envia al VIM#1 la solicitud para crear las máquinas virtuales instanciadas para el vEPC y reservar los recursos necesarios para dichas VMs.

7) El VIM#2 solicita a la capa de virtualización del NFVI#2 la creación de estas VMs para el vIMS (incluye también la formación de los *virtual links* entre las VNFs usando el SDN Controller presente en la capa de virtualización).

8) El VIM#1 solicita a la capa de virtualización del NFVI#1 la creación de estas VMs para el vEPC (incluye también la formación de los *virtual links* entre las VNFs usando el SDN Controller presente en la capa de virtualización).

9) El VIM#2 reporta al NFVO sobre la creación exitosa de las VMs del vIMS.

10) El VIM#1 reporta al NFVO sobre la creación exitosa de las VMs del vEPC.

11) El NFVO notifica al conjunto de VNFMs que las máquinas virtuales necesarias ya están disponibles para ejecutar las VNFs en ellas.

12) Cada VNFM configura cada una de las VNFs del vEPC y del vIMS con los parámetros necesarios para ofrecer el servicio.

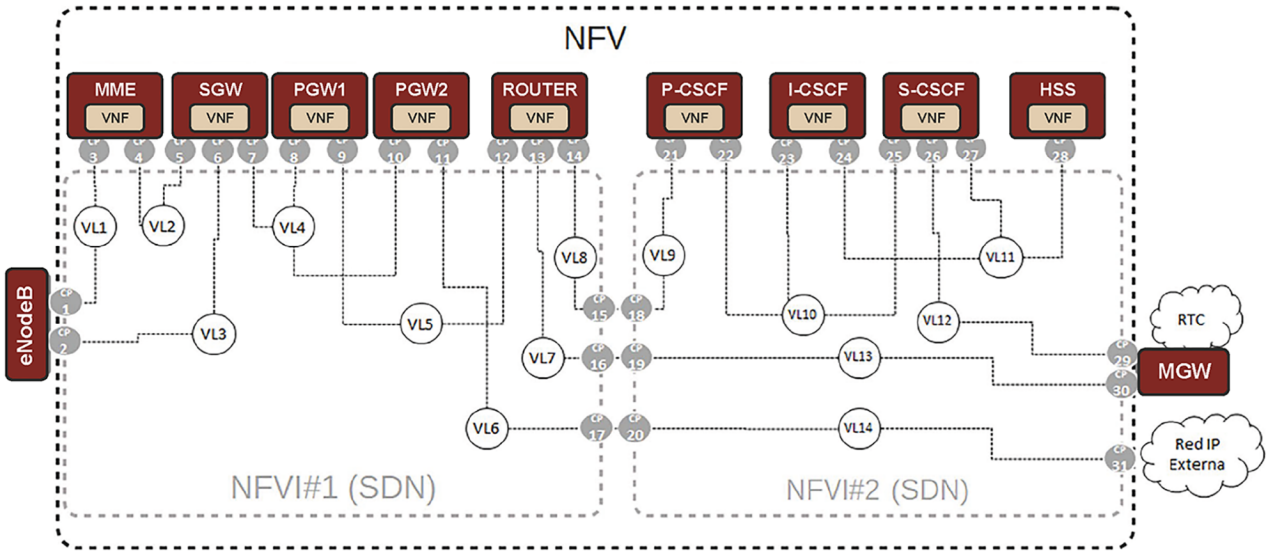
13) Una vez configuradas exitosamente todas las VNFs, el VNFM comunica al NFVO que las VNFs están preparadas, configuradas y disponibles para usarse.

Finalmente, en la Figura 12 vemos el diagrama final de despliegue del vEPC y del vIMS. Podemos observar que cada VNF tiene una serie de puntos de conexión (CP o *connection points*) que ya hemos mencionado antes, pero lo más llamativo son los enlaces virtuales o *virtual links* que interconectan todas

las VNFs y que se implementan usando la tecnología flexible de SDN que ya hemos explicado anteriormente. El *router* VNF también se podría implementar vía SDN.

Cabe destacar que dentro del ámbito del NFV se identifican CPs específicos que están mapeados a puertos físicos como puertos de entrada del tráfico en la zona administrativa del proveedor de servicio que gestiona el ecosistema NFV.

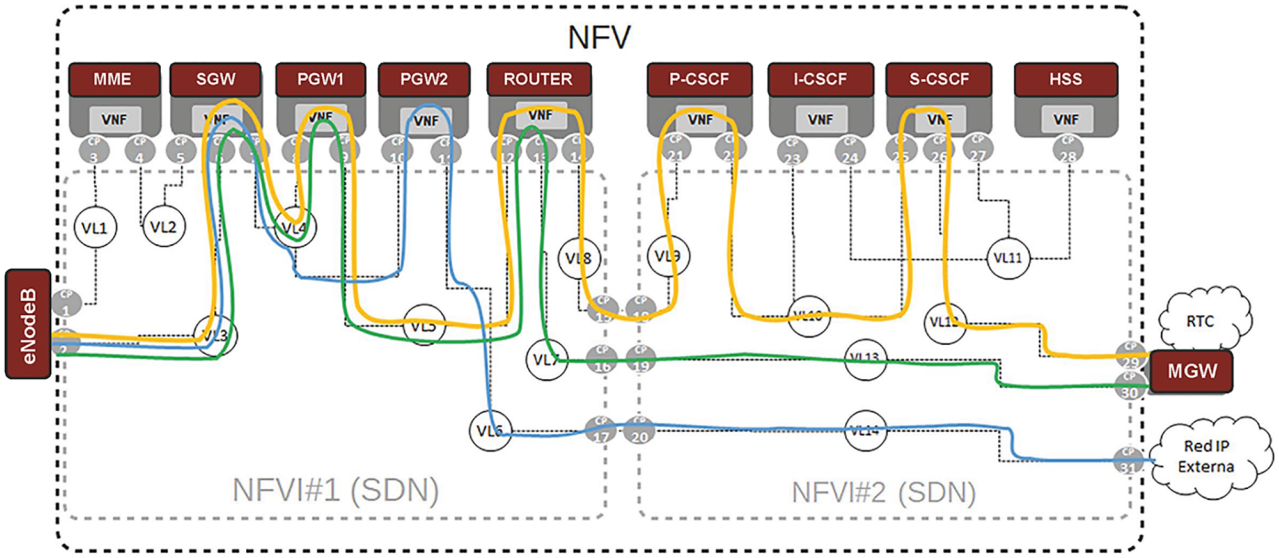
Figura 12. Diagrama de la implementación del vEPC y vIMS con sus *virtual links*.



En la Figura 12 podemos ver los *virtual links* que se configuran para conectar todas las VNFs entre sí. Cada *virtual link* configura un *VNF Forwarding Graph* (VNFFG). En este ejemplo tenemos hasta 14 VNFFGs.

A partir de estos VNFFGs podemos describir varios *Network Forwarding Paths*. Por ejemplo, el tráfico SIP de una llamada IMS a RTC seguiría el siguiente camino indicado de color amarillo en la Figura 13. Para el tráfico de voz RTP se seguiría el NFP de color verde. Finalmente, para el tráfico de Internet sería la línea azul.

Figura 13. Network Forwarding Paths para distintos tráficos.



2.4. Plataformas existentes que implementan NFV

Hay proveedores de servicio que ya han optado por realizar despliegues usando NFV. Es interesante ver qué opciones de implementación hay disponibles en el mercado a la hora de tomar decisiones de inversión en la provisión futura de servicios. Las plataformas de NFV líderes son OpenStack y vCloud NFV de VMware.

OpenStack

Es un *software* gratuito de código abierto que ha sido desarrollado por la colaboración entre la NASA y RackSpace. Es actualmente el proyecto de código abierto para la gestión de plataformas *cloud* más activo hoy en día. OpenStack es un set de herramientas *software* de código abierto para implementar y gestionar arquitecturas de *cloud computing*. Habilita a los proveedores de servicio la provisión y orquestación de clústeres de recursos de centros de datos. Una empresa que quiera implementar su propia plataforma NFV puede internamente intentar montarla con sus propios recursos, aunque existen numerosas empresas que comercializan su propia implementación de OpenStack entre los que podemos encontrar: Red Hat, Dell, Hewlett Packard Enterprise, Ericsson, VMware y Mirantis.

¿Quién ha implementado OpenStack hasta hoy?

Entre los proveedores de servicios que han implementado OpenStack para la infraestructura NFV están: AT&T, Verizon, XO Communications, Comcast, Time Warner Cable, NTT, SK Telecom, China Mobile Ltd., China Unicom, Deutsche Telekom, Swisscom, British Telecom Group y Telefónica.

VMware vCloud NFV

VMware es el proveedor líder de *software* de virtualización para grandes empresas incluyendo departamentos TIC internos de los proveedores de servicios más importantes. vCloud NFV de VMware es una plataforma de infraestructura modular e integrada, la cual está optimizada para el despliegue de NFV en

¿Quién ha implementado vCloud hasta hoy?

VMware tiene más de 80 despliegues de NFV con más de 45 proveedores de servicio.

proveedores de servicio. Además, posee una plataforma OpenStack integrada que habilita a los proveedores de servicio a desplegar rápidamente aplicaciones NFV sobre plataformas OpenStack.

Resumen

La nueva generación de servicios que las redes 5G están introduciendo tienen una serie de requisitos de QoS cada vez más heterogéneos, pero, además, aquellas empresas que provean dichos servicios deberán ser capaces de adaptarse de manera dinámica y rápida a los cambios de demanda que se puedan producir sin que repercuta en altos costes de despliegue y operatividad.

En la arquitectura tradicional de las redes que se han implementado hasta ahora la inteligencia y conocimiento de la tipología se han distribuido a lo largo de todos los elementos físicos que conforman la red. Ello conlleva cierta rigidez y altos costes de operatividad (horas dedicadas por ingenieros) a la hora de adaptar la red a cambios rápidos de topología y a cambios en el dimensionamiento de la red para adaptarse a picos de demanda de un servicio.

Las *Software Defined Networks* introducen un nuevo paradigma donde se propone un modelo totalmente centralizado donde la inteligencia y conocimiento de toda la topología de la red se concentra en un solo punto (SDN Controller) mientras se controlan remotamente todos los dispositivos de red que conforman la capa de infraestructura y que procesan los paquetes (*switches*/*routers*).

Este desacoplo entre la capa de datos y la capa de control es posible gracias al SDN Controller, que actúa como un auténtico sistema operativo de red, y las *Southbound APIs* (SB APIs) que interconectan el SDN Controller y los *switches* que conforman la capa de infraestructura. Entre los SDN Controllers destacan plataformas de código abierto como *Opendaylight* y *OpenNetworking Operating System* (ONOS). A nivel de SB APIs destacan sobretodo *Openflow,* NETCONF y *Open Virtual Switch Database* (OVSDB).

Sobre estos SDN Controllers se pueden ejecutar aplicaciones que implementen funcionalidades tan básicas como un *router* o un *firewall*, a elementos más complejos como priorizadores de paquetes. Sea cual sea la aplicación, ésta deberá interactuar con el SDN Controller a través de las *Nourthbound API* que ofrezca.

No obstante, para que un proveedor de servicios pueda disponer de una infraestructura flexible y adaptable a cambios de modelos de negocio o a incrementos o decrementos repentinos de demanda necesita ejecutar las aplicaciones (que son las que dan valor añadido e inteligencia al servicio) en un entorno completamente virtualizado donde se pueda agilizar al máximo la asignación a dichas aplicaciones de recursos *hardware* a nivel de computación, memoria, almacenaje y red.

La ETSI ha dedicado esfuerzos para especificar el ecosistema que define la *Network Functions Virtualization* (NFV) que en definitiva es aplicar la tecnología de *cloud computing* a la provisión de servicios de telecomunicaciones.

Las tecnologías SDN y NFV se complementan a la hora de proporcionar servicios ya que la primera virtualiza los recursos de red y la segunda ofrece una plataforma de gestión y orquestación para automatizar la creación e interconexión de máquinas virtuales para ejecutar los servidores o funciones en forma de *Virtual Network Functions* (VNFs). La interconexión de VNFs para proporcionar un servicio se llama *service chaining*. Esta plataforma posibilita por ejemplo virtualizar todo un EPC (vEPC) o un núcleo IMS (vIMS).

Hay dos plataformas que implementan todo el ecosistema que especifica NFV: OpenStack (plataforma de código abierto de referencia) y vCLoud de VMware.

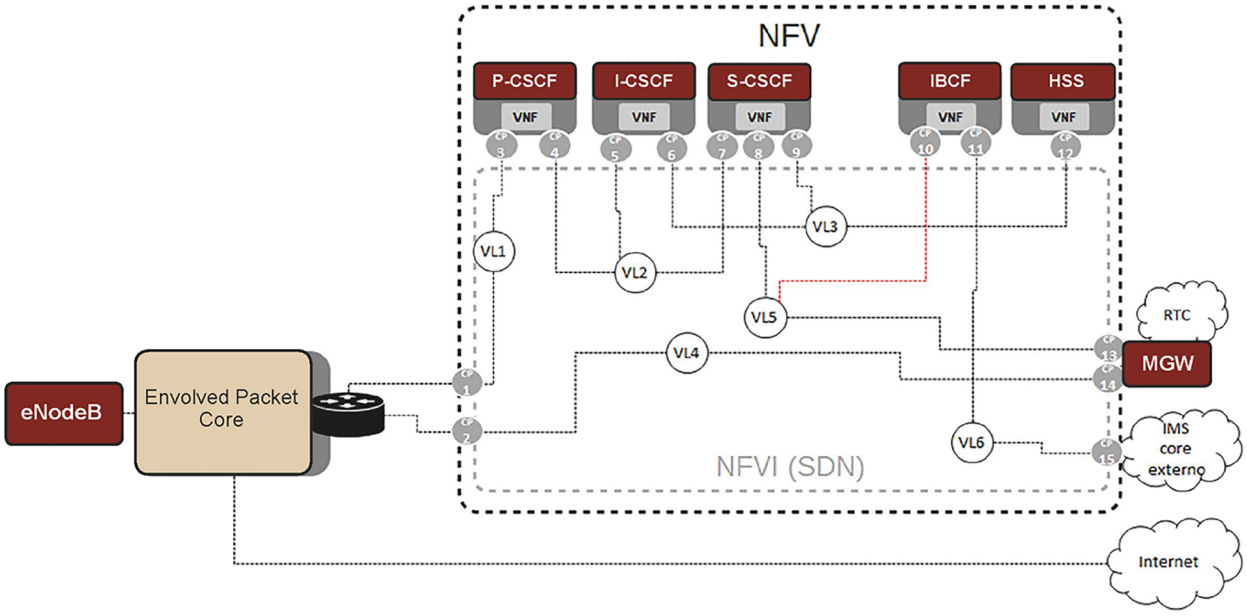
Ejercicios de autoevaluación

1. Un operador de red ha implementado un núcleo IMS virtualizado siguiendo la arquitectura de NFV. Este núcleo IMS está compuesto por los siguientes elementos: un P-CSCF VNF, un I-CSCF VNF, un S-CSCF VNF y un HSS VNF.

El operador habilita el acceso a su núcleo IMS (señalización SIP) desde el EPC a través de un CP específico que está mapeado a un puerto físico de uno de lo *switches* que el operador dispone en su dominio NFVI. Por otro lado, ha habilitado la interconexión con un elemento de red físico, la MGW que interconecta el núcleo con redes heredadas (RTC). Para ello se han habilitado dos CPs dedicados para señalización IMS y para el tráfico RTP.

Este operador quiere implementar la interconexión con otro operador vía señalización IMS y por eso ha incorporado una VNF nueva: IBCF. Con este bloque pretende sacar todo el tráfico SIP hacia al operador externo vía el VL6. El diagrama final se puede ver en la Figura 14.

Figura 14. vIMS Ejercicio 1.



Contestad a las siguientes preguntas:

a) Al intentar establecer una llamada de pruebas con este operador externo ven que la llamada no se establece y lo primero que sospechan es una incorrecta interconexión de la nueva VNF introducida, la VNF IBCF. ¿Observas en la figura algo que justifique que no funcione? ¿Qué corrección propones?

b) Asumiendo que la corrección de la pregunta anterior se ha llevado a cabo correctamente, dibuja el NFP (*Network Forwarding Paths*) para un registro IMS (incluyendo el tráfico SIP y DIAMETER).

c) Si el SDN Controller para implementar la capa de red virtualizada de la NFVI es OpenDaylight y la implementación del vIMS se hace siguiendo OpenStack, ¿Como se llama la *Northbound API* que ofrece este SDN Controller para poder establecer los *virtual links* en la capa de virtualización de red?

d) Las VNFs de P-CSCF, I-CSCF y S-CSCF se ejecutan en una VM cada una por separado, pero usando los mismos recursos *hardware* (es decir, el mismo servidor de propósito general). ¿Qué VL del diagrama se podría implementar sin necesidad de utilizar un recurso de red físico?

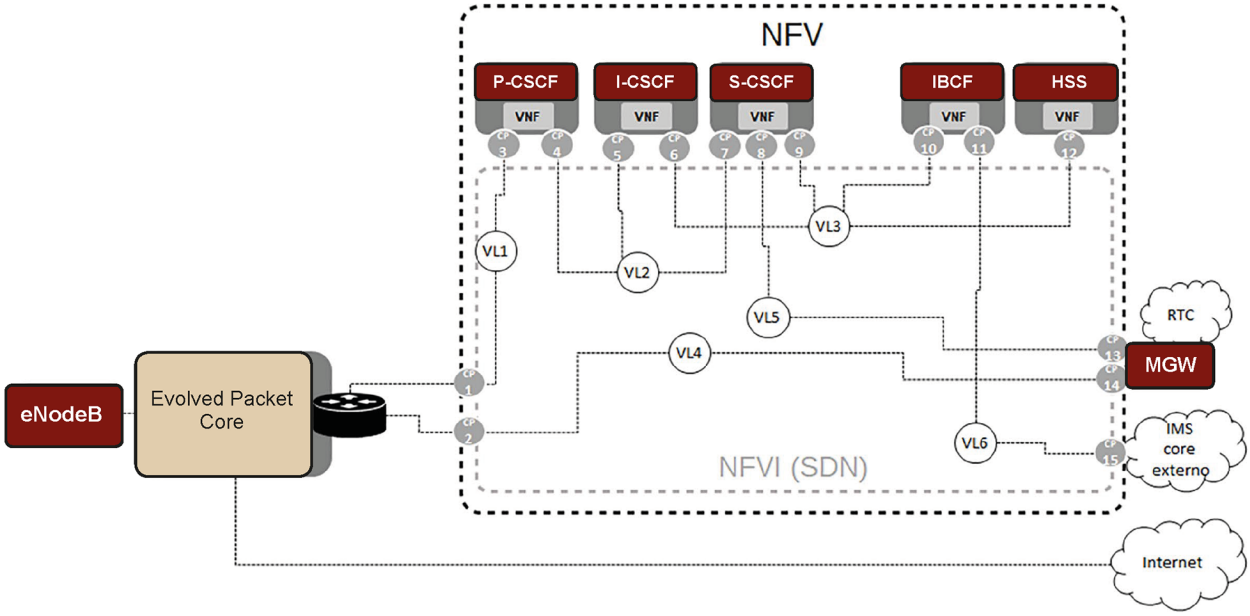
e) ¿Qué *switch* puede implementar este VL?

Solucionario

1.

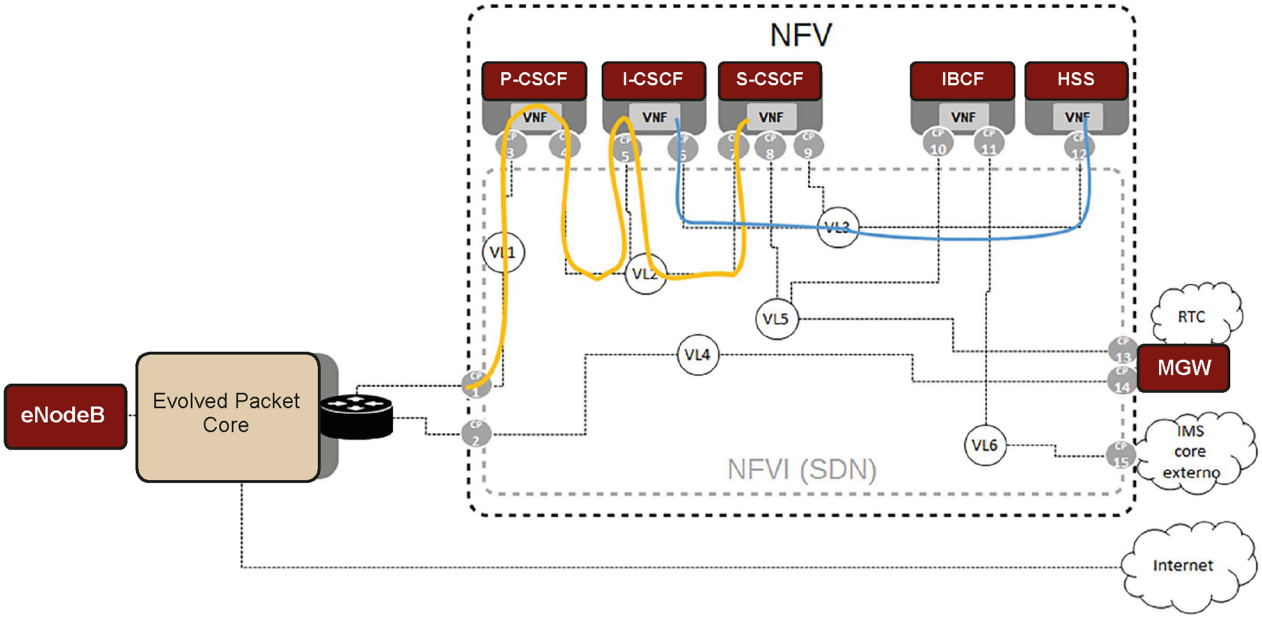
a) El problema que se observa es la interconexión entre el CP10 y el VL3, el cual sirve para interconectar los elementos del núcleo IMS que usan Diameter con el HSS. Es como si hubiésemos aislado al IBCF del resto del núcleo IMS. La solución propuesta es la de interconectar el CP10 con el VL5, que es la que interconecta el S-CSCF con la red externa (vía el MGW), como muestra la Figura 15.

Figura 15. Solución Ejercicio 1a.



b) La figura 16 muestra los dos NFPs. El amarillo es el NFP de SIP y el azul el de Diameter para un registro IMS

Figura 16. Solución Ejercicio 1b.



c) Neutron.

d) El VL2.

e) OVS (*open vSwitch*) ya que se puede implementar exclusivamente vía *software*.

Glosario

5G son las siglas utilizadas para referirse a la quinta generación de tecnologías de telefonía móvil. Es la sucesora de la tecnología 4G.

API *Application Programming Interface*. Conjunto de funcionalidades de programación estandarizadas para desarrollar una funcionalidad común en otras aplicaciones heterogéneas.

ASN.1 *Abstract Syntax Notation One* es una norma para representar datos independientemente de la máquina que se esté usando y sus formas de representación internas. Es un protocolo de nivel de presentación en el modelo OSI. El protocolo SNMP usa el ASN.1 para representar sus objetos gestionables.

CAM *Content-Addressable Memory* es un tipo de memoria de computador empleada en determinadas aplicaciones que requieren velocidades de búsqueda muy elevadas.

Diameter Evolución del protocolo RADIUS para el desarrollo de aplicaciones de AAA.

EMS *Element Management System* proporciona la gestión de red de las funciones de red virtualizadas (VNF) y elementos físicos de red.

ETSI La *European Telecommunications Standards* Institute es una organización de estandarización de la industria de las telecomunicaciones (fabricantes de equipos y operadores de redes) de Europa, con proyección mundial. http://www.etsi.org

FCAPS es el modelo y *framework* de red de gestión de telecomunicaciones de ISO para la gestión de redes. FCAPS es un acrónimo de *Fault, Configuration, Accounting, Performance, Security* (Falla, Configuración, Contabilidad, Desempeño, Seguridad) que son las categorías en las cuales el modelo ISO define las tareas de gestión de redes.

HSS *Home Subscriber Server*. Base de datos que almacena la información de suscripción de un usuario junto con información de autenticación y autorización a nivel de servicio (modelo de referencia del 3GPP).

I-CSCF *Interrogating Call Session Control Function*. Componente del núcleo IMS que ejerce de elemento de encaminador de la señalización SIP hacia el S-CSCF correcto dentro de su mismo dominio. Es un elemento definido por el 3GPP.

IMS El IP Multimedia Subsystem es el estándar definido por el 3GPP para la provisión de servicios multimedia en telefonía móvil basado en los protocolos definidos por IETF, como SIP, RTP o Diameter.

MIB es un tipo de base de datos que contiene información jerárquica, estructurada en forma de árbol, de todos los parámetros gestionables en cada dispositivo gestionado de una red de comunicaciones.

NBI *Northbound Interface* es la interfaz que una entidad usa para publicar a otra entidad los servicios programáticos que ofrece.

NETCONF *Network Configuration Protocol* es un protocolo de gestión de red desarrollado y estandarizado por el IETF que proporciona mecanismos para instalar, manipular y eliminar la configuración de los dispositivos de red.

NFV *Network Function Virtualization* es un enfoque de red en evolución que permite la sustitución de dispositivos de *hardware* dedicados y costosos tales como *routers*, *switches*, firewalls y equilibradores de carga con dispositivos de red basados en *software* que se ejecutan como máquinas virtuales en servidores estándares de la industria.

NFV MANO *Network Functions Virtualization Management and Orchestration* es un marco arquitectónico para administrar y orquestar funciones de red virtualizadas (VNF) y otros componentes de *software*.

NFVI *Network Function Virtualization Infrastructure* define estándares para recursos de computación, almacenamiento y redes que se pueden usar para construir funciones de red virtualizadas. Es un componente clave de la arquitectura NFV que describe los componentes de *hardware* y *software* en los que se crean las redes virtuales.

NFVO *Network Function Virtualization Orchestrator* es un componente clave del marco arquitectónico NFV-MANO, que realiza orquestación de recursos y orquestación de servicios de red, así como otras funciones. Es un componente central de una solución basada en NFV. Une diferentes funciones para crear un servicio de extremo a extremo coordinado.

OF Openflow es considerado uno de los primeros estándares de redes definidas por *software* (SDN). Definió originalmente el protocolo de comunicación en entornos SDN que permite que el controlador SDN interactúe directamente con la capa de datos de los dispositivos de red, como *switches* y *routers*, tanto físicos como virtuales (basados en hipervisor), para adaptarse mejor a los requisitos comerciales cambiantes.

OpenStack Es un *software* gratuito de código abierto que ha sido desarrollado para implementar NFV. Es actualmente el proyecto de código abierto para la gestión de plataformas *cloud* más activo hoy en dia.

OSS/BSS *Operational Support System/Business Support System*. OSS es un conjunto de programas que ayudan a un proveedor de servicios de comunicaciones a monitorear, controlar, analizar y administrar un teléfono o una red de computadoras. BSS son los componentes que un proveedor de servicios de telecomunicaciones (o compañía de telecomunicaciones) utiliza para ejecutar sus operaciones comerciales hacia los clientes.

OVSDB *Open vSwitch Database* es un protocolo de configuración de OpenFlow que está diseñado para administrar implementaciones de Open vSwitch.

OVS *Open vSwitch* es un *software* de código abierto, diseñado para ser utilizado como un *switch* virtual en entornos de servidores virtualizados. Es el encargado de reenviar el tráfico entre diferentes máquinas virtuales (VMs) en el mismo *host* físico y también reenviar el tráfico entre las máquinas virtuales y la red física.

P-CSCF *Proxy Call Session Control Function*. Componente del núcleo IMS que ejerce de elemento fronterizo con el equipo de usuario a nivel de señalización SIP (IMS). Es un elemento definido por el 3GPP.

RTC Red Telefónica Conmutada.

RTP *Real Time Protocol*. Protocolo basado en UDP para la transmisión de flujos multimedia (audio, vídeo) en tiempo real.

S-CSCF *Serving Call Session Control Function*. Componente del núcleo IMS que ejerce de registrador del usuario a nivel de capa de control de servicio y de encaminador de la señalización hacia otros elementos que finalicen la llamada dentro del mismo dominio o de otro distinto. Es un elemento definido por el 3GPP.

SBI *Southbound Interface* es una interfaz usada por una entidad para programar a un tercer dispositivo de manera remota. En el contexto de SDN el SBI es la interfaz que usa el SDN Controller para controlar la capa de infraestructura. Ejemplos: Openflow y NETCONF.

SDN *Software Defined Network* utilizan *software* en lugar de dispositivos especializados para el aprovisionamiento y la gestión de los servicios de las aplicaciones y de la red, lo que permite la movilidad y el despliegue de aplicaciones programables, escalables y bajo demanda. SDN supervisa de manera inteligente y adapta automáticamente la red para proveer un mejor servicio en las condiciones de cada momento.

SIP El *Session Initiation Protocol* es un protocolo definido por el IETF para el establecimiento y negociación de sesiones de servicios multimedia.

SNMP *Simple Network Management Protocol* es un protocolo de la capa de aplicación que facilita el intercambio de información de administración entre dispositivos de red. Los dispositivos que normalmente soportan SNMP incluyen *routers*, *switches*, servidores, estaciones de trabajo, impresoras, bastidores de módem y muchos más.

VIM *Virtualized Infrastructure Manager* es un elemento del entorno arquitectural NFV MANO y es el responsable de controlar y administrar los recursos informáticos, de almacenamiento y de red de la infraestructura NFV (NFVI), generalmente dentro del dominio de infraestructura de un operador.

VM *Virtual Machine* es un sistema operativo (SO) o entorno de aplicación que está instalado en un *software* que imita el *hardware* dedicado.

VNF *Virtual Network Functions* son tareas virtualizadas llevadas a cabo anteriormente por *hardware* propietario y exclusivo. Las VNF mueven las funciones de red individuales de los dispositivos de *hardware* dedicados al *software* que se ejecuta en *hardware* de propósito general.

VNFM *Virtual Network Function Managers* gestionan le VNF durante todo su ciclo de vida y son fundamentales para escalar, operaciones de modificación, agregar nuevos recursos y comunicar los estados de las VNF a otros bloques funcionales en la arquitectura NFV-MANO.

YANG *Yet Another Next Generation* es un lenguaje de modelado de datos para la definición de datos enviados a través del protocolo de configuración de red NETCONF.

Bibliografía

Chayapathi, R.; Hassan, S. F.; Shah, P. (noviembre 2016). *Network Functions Virtualization (NFV) with a Touch of SDN*. Addison-Wesley Professional.

Kahn, F. (marzo 2015). *A Cheat Sheet for Understanding "NFV Architecture"*. https://www.telcocloudbridge.com/blog/a-cheat-sheet-for-understanding-nfv-architecture/

Kahn, F. (abril 2015). *A Beginner's Guide to NFV Management & Orchestration (MANO)*. https://www.telcocloudbridge.com/blog/a-beginners-guide-to-nfv-management-orchestration-mano/

Rouse, M. (noviembre 2017). *Virtual network functions (VNF)*. https://searchsdn.techtarget.com/definition/virtual-network-functions

Rouse, M. (diciembre 2017). *REST (REpresentational State Transfer)*. https://searchmicroservices.techtarget.com/definition/REST-representational-state-transfer

Toghraee, R. (mayo 2017). *Learning Opendaylight*. Packt Publishing.