Memoria del Trabajo Final de Grado

Estudio de las tecnologías SDN y NFV

Área: Administración de redes y sistemas operativos
Autor: Francisco José Ibáñez García
Titulación: Grado Ingeniería Informática
Consultor: Manuel Jesús Mendoza Flores
Fecha: 08 de enero de 2016

Dedicatoria y agradecimientos

Sois muchos y muchas a los que debería agradeceros vuestra ayuda y soporte, por eso quiero deciros a todos: MUCHAS GRACIAS!!!

No obstante, me gustaría hacer una mención especial a ciertas personas.
Me gustaría agradecer a Manel el darme la oportunidad de realizar este proyecto y ofrecerme su ayuda y experiencia a la hora de hacerlo.

A Neus, mi tutora, y a todos/as los profesores y consultores que me han ayudado durante mi paso por la UOC.

A todos/as los que estuvisteis a mi lado y me animasteis en esos malos momentos en los que pensé que no merecía la pena seguir, vosotros ya sabéis a que me refiero. Sin vuestro apoyo no habría salido adelante.

Por supuesto a Núria, por su comprensión y apoyo constante durante todo este tiempo, especialmente en estos últimos meses en los que he realizado este proyecto. Gracias por estar siempre a mi lado.

Y por último a mis padres, que se han esforzado toda su vida para educarnos a mi hermano y a mí de la mejor manera posible y dárnoslo todo sin pedir nunca nada a cambio.

Gracias a todos!!!

Resumen

Durante un gran periodo de tiempo las redes prácticamente no han sufrido cambios en sus arquitecturas. Pero, paralelamente, la tecnología ha cambiado y evolucionado con la aparición y consolidación de nuevas tendencias, como la aparición de una gran cantidad de dispositivos que se conectan a la red (Internet of Things -IoT), aplicaciones en tiempo real, el video streaming, la masificación de las redes sociales, cloud computing, nuevos servicios,…

Debido a esto, las redes se han visto obligadas a soportar unas exigencias para las que no fueron diseñadas originalmente. Esto hace que surja la necesidad de cambiar la forma de comunicación de las redes, en la cual se le proporcione mayor inteligencia a la misma.

Las tecnologías que se estudian en este proyecto (SDN y NFV) ayudan a adecuar las redes a estas nuevas necesidades alejando las redes del hardware para acercarlas al software, simplificando y facilitando su evolución y su gestión.

Las Redes Definidas por Software (SDN, del inglés Software Defined Network) es una nueva tecnología que tiene como objetivo principal simplificar la creación y gestión de las redes. Para conseguirlo separa el plano de control de la red del plano de reenvío. De este modo se consigue controlar todo el comportamiento de la red mediante un elemento lógico centralizado (software), llamado controlador. Esta separación de los planos proporciona a los usuarios una abstracción lógica de los recursos de red subyacentes.

OpenFlow es el primer estándar SDN y un elemento vital en una arquitectura de red SDN. OpenFlow permite la programación remota del plano de control.

El resultado es lograr una arquitectura que permite a los administradores de red tener el control total del funcionamiento de la red a través del despliegue de software que controla y automatiza el comportamiento de la misma.

Por otro lado, en la actualidad, muchos servicios se soportan en sistemas independientes de servicios, es decir, cada servicio incluye su propio hardware, software y equipo de operaciones. Con NFV, los proveedores de servicios pueden desplegar un modelo más horizontal en el que la plataforma NFV (con sus recursos de red, almacenamiento y computación) es una capa común que no es necesario duplicar para cada servicio.

Virtualización de las Funciones de la Red (NFV, del inglés Network Functions Virtualization) permite implementar funciones de red mediante software, en lugar de hardware, lo que permite eliminar la necesidad de dispositivos de red dedicados, tales como routers, switches y firewalls. Trasladar las funciones de red de hardware específico hacia servidores básicos ayuda a los proveedores a evitar una sobrecompra de equipos y la subutilización de recursos.

A pesar de que SDN y NFC son tecnologías diferentes y pueden perfectamente existir la una sin la otra, son conceptos muy relacionados entre sí y ambas pueden ser complementarias. De hecho, con la combinación de ambas tecnologías se consigue un mayor potencial y una mayor adaptación a las nuevas necesidades tecnológicas.

La creación de estándares orientados a estas tecnologías, especialmente los de código abierto (como por ejemplo los proyectos OpenFlow, OpenDayLight y OpenStack), ha supuesto para los desarrolladores y proveedores de servicios una “liberación”. Esto se debe a que estos estándares o protocolos les permiten desarrollar nuevos servicios de red basados en aplicaciones, liberándose del software propietario de los dispositivos de red convencionales que limitan ese desarrollo.

A pesar de que las tecnologías SDN y NFV son relativamente nuevas, están generando muchas expectativas ante las cuales el mercado del networking se está moviendo. De hecho, los grandes proveedores (y algunos no tan grandes) ya han lanzado algunas soluciones, con el objetivo de conseguir hacerse un hueco dentro de este mercado emergente.

Índice General

Índice de Figuras y Tablas

Índice de figuras

Índice de tablas

1. Capítulo 1. Introducción.

1.1 Justificación del proyecto.

Ha habido un gran periodo de tiempo en el que las redes prácticamente no han sufrido cambios en sus arquitecturas. Pero, paralelamente, en los últimos años la tecnología ha cambiado considerablemente con la aparición y consolidación de nuevas tendencias, como la aparición de una gran cantidad de dispositivos que se conectan a la red (Internet of Things -IoT), grandes volúmenes de datos, cloud computing, nuevas aplicaciones y servicios,…
Debido a esto, las redes se ven obligadas a soportar unas exigencias para las que no fueron diseñadas originalmente, lo que hace que tengan la necesidad de modificar su enfoque.
Las tecnologías que se estudian en este proyecto (SDN y NFV) ayudan a adecuar las redes a estas nuevas necesidades. Concretamente ayudan a alejar las redes del hardware para acercarlas al software, simplificando y facilitando su evolución y su gestión.
Por otro lado, el hecho de que estas tecnologías sean bastante recientes hace que sean algo desconocidas. No obstante grandes empresas ya están empezando a adoptarlas a la hora de implementar sus redes.

Todo esto justifica que se realice un estudio de estas tecnologías como el que se pretende en este proyecto.

1.2 Objetivos del proyecto.

El principal objetivo para la realización de este proyecto es estudiar y conocer las tecnologías SDN y NFV con sus ventajas y sus problemas, de tal forma que se comprendan y entiendan ambos conceptos.
Pero dicho esto, para conseguir un estudio completo, hay que conseguir alcanzar otra serie de objetivos que complementen el estudio y así conseguir un estudio completo que aporte una visión general de estas tecnologías y una mejor comprensión de las mismas.
A continuación se dan dos listas mas detalladas de los objetivos, la primera con los objetivos principales y la segunda con los objetivos parciales (necesarios para conseguir los principales).
Los principales objetivos de este proyecto son:

- Estudiar y analizar las tecnologías SDN y NFV.
- Realizar una comparativa de las tecnologías SDN y NFV.
- Crear un escenario SDN virtual.
- Implementar dos casos de uso.
- Obtener una serie de conclusiones.

Los objetivos parciales para lograr estos objetivos principales son los siguientes:

- Conocer la historia de las tecnologías SDN y NFV (su origen, evolución y situación actual).
- Conocer que fabricantes y empresas están implicados y que hacen.
- Analizar los nuevos estándares e implementaciones que están surgiendo.
- Hacer unos scripts a modo de ejemplo, sobre un escenario SDN virtual, que implemente algunas funcionalidades.

1.3 Enfoque y método utilizado.

El principal enfoque de este proyecto es el estudio de las tecnologías SDN y NFV. Es decir, recabar la mayor cantidad de información posible sobre estas tecnologías para generar un informe o memoria.
Teniendo en cuenta que los conocimientos iniciales son prácticamente nulos, se podría decir que es un enfoque "investigador".

La metodología utilizada ha consistido principalmente en la búsqueda de información de estas tecnologías. Una vez se ha tenido la información necesaria se ha procedido a su análisis, clasificación, selección,… para posteriormente redactar con ella un informe o memoria.

Además, este informe se complementa con dos ejemplos práctico de dos casos de uso implementados sobre un escenario SDN virtual.

1.4 Planificación del proyecto.

La planificación se ha dividido en 2 partes, por un lado un cronograma y por otro un diagrama de Gannt. El primero permite una visión general de las tareas y sus tiempos y el segundo una visión más detallada y precisa.

Cronograma
A continuación se muestra el cronograma del proyecto en forma de tabla.
Este cronograma está detallado por semanas y por lo tanto no se reflejan las horas exactas que se dedican a cada tarea. El objetivo de este cronograma es tener una visión general de la planificación del proyecto y una idea aproximada de los tiempos empleados en cada tarea. Concretamente se pueden observar las tareas que se realizan cada semana y a que partes de la memoria se corresponden estas tareas. También se pueden observar las entregas intermedias previstas.

TFG			
	Semana	**Actividad**	**Memoria**
1	16-sep al 20-sep	Elección y definición del proyecto	Capítulo 1
2	21-sep al 27-sep	Concreción de los puntos a desarrollar	Capítulo 1
3	28-sep al 04-oct	Realización y entrega PAC1: Plan de trabajo	Índice y capítulo 1
4	05-oct al 11-oct	Tarea 1:Análisis y estudio de la tecnología SDN	Capítulo 2
5	12-oct al 18-oct	Tarea 1:Análisis y estudio de la tecnología NFV	Capítulo 2
6	19-oct al 25-oct	Tarea 2: Realización de una comparativa	Capítulo 3
7	26-oct al 01-nov	Tarea 2: Realización de una comparativa	Capítulo 3
8	02-nov al 08-nov	Entrega PAC2: Entre el 40% y el 60%	Capítulos 1, 2 y 3
9	09-nov al 15-nov	Tarea 3: Creación de un escenario	Capítulo 4
10	16-nov al 22-nov	Tarea 3: Creación de un escenario	Capítulo 4
11	23-nov al 29-nov	Tarea 4: Implementación caso de uso 1	Capítulo 5
12	30-nov al 06-dic	Tarea 4: Implementación caso de uso 2	Capítulo 5
13	07-dic al 13-dic	Entrega PAC3: Entre el 80% y el 90%.	Capítulos 4 y 5
14	14-dic al 20-dic	Conclusiones	Capítulo 7
15	21-dic al 27-dic	Resto de apartados	Capítulo 6
16	28-dic al 08-ene	Entrega del TFG y Presentación	100% memoria

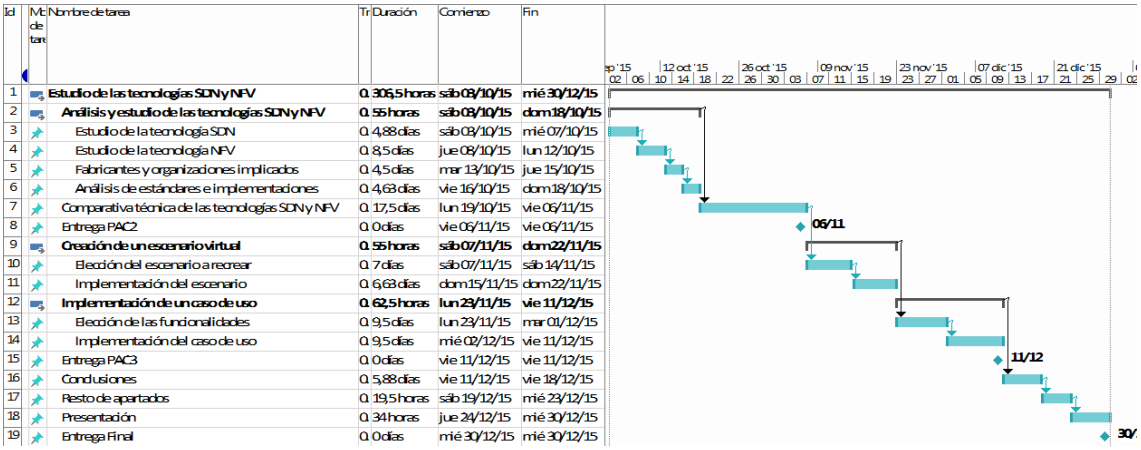
Tabla 1.1 Cronograma del proyecto

Diagrama de Gannt

A continuación se muestra una imagen del diagrama de Gannt correspondiente al proyecto. En él se muestran todas las tareas del proyecto de forma detallada definiendo para cada una de ellas un periodo para su realización y las horas reales empleadas.

Además, se han colocado varios hitos intermedios que se corresponden con las entregas intermedias previstas.

Para la realización de la planificación se han definido todos los días como laborables, excepto los jueves. Y se ha asignado una dedicación media de 4 horas diarias, que es lo que se calcula que se podrá dedicar normalmente.



Id	Modo de tarea	Nombre de tarea	Trabajo	Duración	Comienzo	Fin
1		**Estudio de las tecnologías SDN y NFV**	0	306,5 horas	sáb 03/10/15	mié 30/12/15
2		**Análisis y estudio de las tecnologías SDN y NFV**	0	55 horas	sáb 03/10/15	dom 18/10/15
3		Estudio de la tecnología SDN	0	4,88 días	sáb 03/10/15	mié 07/10/15
4		Estudio de la tecnología NFV	0	8,5 días	jue 08/10/15	lun 12/10/15
5		Fabricantes y organizaciones implicados	0	4,5 días	mar 13/10/15	jue 15/10/15
6		Análisis de estándares e implementaciones	0	4,63 días	vie 16/10/15	dom 18/10/15
7		Comparativa técnica de las tecnologías SDN y NFV	0	17,5 días	lun 19/10/15	vie 06/11/15
8		Entrega PAC2	0	0 días	vie 06/11/15	vie 06/11/15
9		**Creación de un escenario virtual**	0	55 horas	sáb 07/11/15	dom 22/11/15
10		Elección del escenario a recrear	0	7 días	sáb 07/11/15	sáb 14/11/15
11		Implementación del escenario	0	6,63 días	dom 15/11/15	dom 22/11/15
12		**Implementación de un caso de uso**	0	62,5 horas	lun 23/11/15	vie 11/12/15
13		Elección de las funcionalidades	0	9,5 días	lun 23/11/15	mar 01/12/15
14		Implementación del caso de uso	0	9,5 días	mié 02/12/15	vie 11/12/15
15		Entrega PAC3	0	0 días	vie 11/12/15	vie 11/12/15
16		Conclusiones	0	5,88 días	vie 11/12/15	vie 18/12/15
17		Resto de apartados	0	19,5 horas	sáb 19/12/15	mié 23/12/15
18		Presentación	0	34 horas	jue 24/12/15	mié 30/12/15
19		Entrega Final	0	0 días	mié 30/12/15	mié 30/12/15

Figura 1.1 Diagrama de Gannt inicial del proyecto

No obstante, durante la ejecución del proyecto se han realizado algunos cambios que han hecho que se tenga que modificar el diagrama de Gannt original. En la figura 1.2 se puede observar el diagrama de Gannt actual, que incluye estas modificaciones. El principal cambio es la implementación de 2 casos de uso en lugar de 1, la modificación de la fecha de entrega y la variación de las horas reales utilizadas respecto a las previstas inicialmente.

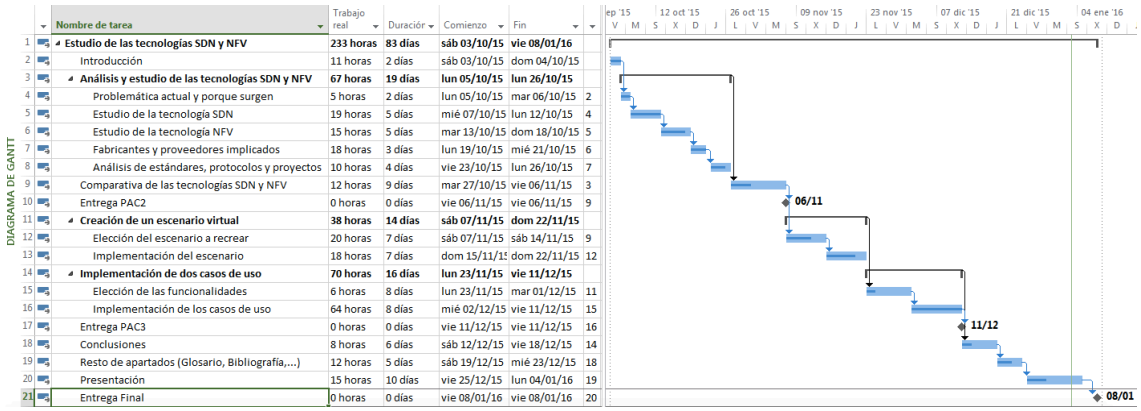


DIAGRAMA DE GANTT

	Nombre de tarea	Trabajo real	Duración	Comienzo	Fin	
1	**Estudio de las tecnologías SDN y NFV**	**233 horas**	**83 días**	**sáb 03/10/15**	**vie 08/01/16**	
2	Introducción	11 horas	2 días	sáb 03/10/15	dom 04/10/15	
3	**Análisis y estudio de las tecnologías SDN y NFV**	**67 horas**	**19 días**	**lun 05/10/15**	**lun 26/10/15**	
4	Problemática actual y porque surgen	5 horas	2 días	lun 05/10/15	mar 06/10/15	2
5	Estudio de la tecnología SDN	19 horas	5 días	mié 07/10/15	lun 12/10/15	4
6	Estudio de la tecnología NFV	15 horas	5 días	mar 13/10/15	dom 18/10/15	5
7	Fabricantes y proveedores implicados	18 horas	3 días	lun 19/10/15	mié 21/10/15	6
8	Análisis de estándares, protocolos y proyectos	10 horas	4 días	vie 23/10/15	lun 26/10/15	7
9	Comparativa de las tecnologías SDN y NFV	12 horas	9 días	mar 27/10/15	vie 06/11/15	3
10	Entrega PAC2	0 horas	0 días	vie 06/11/15	vie 06/11/15	9
11	**Creación de un escenario virtual**	**38 horas**	**14 días**	**sáb 07/11/15**	**dom 22/11/15**	
12	Elección del escenario a recrear	20 horas	7 días	sáb 07/11/15	sáb 14/11/15	9
13	Implementación del escenario	18 horas	7 días	dom 15/11/15	dom 22/11/15	12
14	**Implementación de dos casos de uso**	**70 horas**	**16 días**	**lun 23/11/15**	**vie 11/12/15**	
15	Elección de las funcionalidades	6 horas	8 días	lun 23/11/15	mar 01/12/15	11
16	Implementación de los casos de uso	64 horas	8 días	mié 02/12/15	vie 11/12/15	15
17	Entrega PAC3	0 horas	0 días	vie 11/12/15	vie 11/12/15	16
18	Conclusiones	8 horas	6 días	sáb 12/12/15	vie 18/12/15	14
19	Resto de apartados (Glosario, Bibliografía,...)	12 horas	5 días	sáb 19/12/15	mié 23/12/15	18
20	Presentación	15 horas	10 días	vie 25/12/15	lun 04/01/16	19
21	Entrega Final	0 horas	0 días	vie 08/01/16	vie 08/01/16	20

Figura 1.2 Diagrama de Gannt actual del proyecto

1.5 Productos obtenidos y ámbito de aplicación.

Como ya se ha comentado en los apartados anteriores, este proyecto consiste básicamente en la realización de un estudio de las tecnologías SDN y NFV. Por lo tanto, el resultado (o producto) obtenido de este proyecto no deja de ser esta propia memoria. Como parte de esta memoria también se obtienen algunos scripts que

podrían utilizarse o adaptarse para ejecutarse en otras redes de este tipo.
Por otro lado, este estudio también podría plantearse o enfocarse como una especie de consultoría para conocer y comparar ambas tecnologías.
En cuanto al ámbito de aplicación se refiere, al tratarse de un estudio, no existe un ámbito de aplicación de este proyecto en sí mismo.
No obstante, este proyecto podría tomarse como base de partida para una posible implantación de estas tecnologías.
¿Qué empresas u organismos podrían plantearse la implantación de alguna de estas tecnologías? En cuanto a la implantación de redes virtuales basadas en cualquiera de las dos tecnologías estudiadas en este proyecto, a priori se podría llevar a cabo en cualquier compañía. Pero en la práctica, actualmente la implantación de estas tecnologías está más orientada a las grandes empresas. Especialmente aquellas que han de ofrecer muchos servicios y a muchos usuarios y que por lo tanto cuentan con grandes y complejas redes con muchos equipos y complicadas de gestionar.

1.6 Breve descripción de los otros capítulos de la memoria.

A grandes rasgos, se podría decir que esta memoria está organizada en 3 grandes bloques o apartados, cada uno de ellos con una finalidad diferente:

- El primer bloque englobaría los capítulos 2 (Análisis y estudio de las tecnologías SDN y NFV) y 3 (Comparativa de las tecnologías SDN y NFV).

  La finalidad de este bloque sería el estudio "teórico" de las tecnologías SDN y NFV. Es decir, la recogida de información y posterior análisis de esta.

  Concretamente, en el capítulo 2 se hace el estudio de las tecnologías SDN y NFV. Y en el capítulo 3 se hace una comparativa de estas tecnologías.
- El segundo bloque englobaría los capítulos 4 (Creación de un escenario SDN virtual) y 5 (Implementación de un caso de uso).

  La finalidad de este bloque sería la implementación de un caso práctico, a modo de ejemplo, que permita ver cómo funcionan estas tecnologías y que se puede hacer con ellas.

  Concretamente, en el capítulo 4 se recrea un escenario SDN virtual con Mininet para, posteriormente en el capítulo 5, implementar dos casos de uso prácticos con unos ejemplos de algunas de las funcionalidades de estas tecnologías.
- Y por último, el tercer bloque se correspondería con el capítulo 7 (Conclusiones).

  La finalidad de este capítulo es redactar una serie de conclusiones basadas en todo lo aprendido y analizado en los capítulos anteriores.

Sin olvidar, claro está, otros capítulos como el 6, el glosario, la bibliografía y los anexos.
En el capítulo 6 (Valoración productiva y presupuestaria) se explican los gastos de este estudio.
Posteriormente, en el capítulo correspondiente al Glosario se pretende definir algunos conceptos y siglas que aparecen a lo largo de la memoria.
A continuación, en el capítulo de Bibliografía se hace referencia a todas las fuentes consultadas durante la realización de este proyecto, que mayoritariamente han sido páginas web.
Y por último, en los anexos se añade información complementaria (código de los scripts implementados,…) pero no por ellos menos importante, ya que ayuda a entender el resto de capítulos de la memoria. Simplemente se ha puesto en los anexos para simplificar un poco estos capítulos.

2. Capítulo 2, Análisis y estudio de las tecnologías SDN y NFV.

2.1 Problemática actual y porque surgen estas tecnologías

En este capítulo se pretende explicar que son las tecnologías SDN y NFV, en que consisten, como funcionan,…
Pero para poder llegar a entenderlas bien, antes es importante sentar unas bases y explicar cuál es su punto de partida y porque han surgido [1], [2], [3].

Hace tan sólo algunos años que se empezó a hablar en serio de las tecnologías SDN y NFV ya que parece que pueden aportar cambios importantes en los escenarios actuales de las comunicaciones. Pero nos podríamos preguntar, ¿son necesarios cambios en las redes actuales?. Pues al parecer sí, ya que desde hace algunos años se está poniendo de manifiesto que las actuales redes no permiten, ni facilitan, la aparición de nuevos protocolos ni la implantación de nuevos servicios. Es decir, no evolucionan todo lo rápido que debieran.
La proliferación de Internet en todo el mundo fue un punto de inflexión ya que aparecen nuevos conceptos como “internet de las cosas” (IoT, por sus siglas en inglés, objetos cotidianos que se conectan a internet: relojes, TV,…) o “Cloud Computing” (permite el acceso remoto a aplicaciones e información sin problemas de disponibilidad y almacenamiento). Ante esta situación, los operadores se ven obligados a intentar estandarizar sus protocolos de comunicaciones y proporcionar permanentemente a sus usuarios servicios y productos teniendo siempre en cuenta la ubicuidad de acceso.
Pero la creación de nuevos protocolos que se adapten a estas nuevas necesidades obliga a realizar cambios en las redes, la mayoría de ellas complejas. Esto conlleva asumir una serie de riesgos que no siempre se asumen pensando que “la red no es prefecta pero funciona”.
No obstante, la evolución tecnológica es imparable y los operadores se ven obligados a crear nuevos overlays por encima de la capa IP (VLAN, túneles,…) para poder poner en marcha nuevos servicios de red. Lo que se podría considerar un inicio de la virtualización de la red.

Desde el punto de vista hardware, esta rápida evolución de la tecnología hace que el ciclo de vida de los equipos de red sea cada vez más corto. Lo que supone un problema para muchos operadores, no sólo a nivel tecnológico, sino también a nivel económico. Ya que se ven obligados a realizar cada vez más frecuentemente fuertes inversiones en renovación de equipos.
Con la finalidad de dar respuesta a estos problemas, aparece un nuevo concepto de red protagonizado por las tecnologías SDN y NFV. Estas tecnologías alejan las redes del hardware para acercarlas al software, con el fin de intentar agilizar y flexibilizar el diseño, desarrollo y posterior gestión de las redes.

Resumiendo, las tecnologías SDN y NFV son el inicio de una nueva tendencia que pretende adaptar las actuales redes a las nuevas necesidades y exigencias tecnológicas. Y todo esto simplificando la gestión de estas redes.
A pesar de que tanto SDN como NFV pueden existir por separado, en muchas ocasiones se implementan juntas ya que se complementan y proporcionan un mayor potencial.

2.2 Estudio de la tecnología SDN. Origen y evolución. En que consiste.

Origen y evolución [1], [3]

SDN son las siglas de Software Defined Networking (redes definidas por software, en español).

El origen de SDN es fruto de la experimentación universitaria y no apareció de forma repentina sino que fue fruto de una serie de innovaciones orientadas a hacer las redes más programables. Esto hace que no haya una fecha exacta de aparición.

Pero se calcula que apareció hace unos 20 años, aproximadamente en el comienzo de lo que hoy conocemos como Internet. Y fue precisamente el éxito de internet lo que hizo que surgiera la necesidad de gestionar y evolucionar las infraestructuras de redes, es decir, hacerlas programables.

A partir de este momento, la historia de SDN se divide en tres etapas:

- **Redes activas** (1995-2000 aprox.).

  Surgieron orientadas hacia el control de la red, poniendo énfasis en una interfaz de programación (API). Permitieron reducir el coste computacional, avanzar en lenguajes de programación y en la tecnología de máquinas virtuales.

  Aunque estas redes no tuvieron un despliegue extendido, ofrecieron contribuciones relacionadas con SDN como funciones programables en la red, virtualización de redes y la visión de una arquitectura unificada en distintos aparatos de red como firewalls, IDSs, NATs, etc, …

- **Separación del plano de control del plano de datos** (2001-2007 aprox.).

  Los fabricantes de hardware empezaron a implementar la lógica de reenvío de paquetes en hardware (plano de datos), separada del plano de control. Y los ISPs empezaron a preocuparse por la gestión sus crecientes redes y poder ofrecer a sus clientes servicios que las hiciesen más seguras (como VPNs).

- **Aparición de la interfaz de programación de aplicaciones de OpenFlow** (2007-2010 aprox.).

  Para abordar la visión de separación de plano de datos y de control, se empezó a investigar en nuevas arquitecturas para el control lógico centralizado. Por ejemplo, en la universidad de Stanford, un grupo de investigadores creó el Clean Slate Program, orientado a la experimentación en redes universitarias más tratables y locales, que dio lugar al protocolo Openflow.

  La adopción de Openflow en las empresas favoreció que estas abrieran sus APIs para permitir a los programadores controlar ciertos comportamientos de reenvío.

Como se puede ver, SDN ha ido evolucionando hasta llegar a la actualidad. En los últimos años SDN ha hecho que el diseño y gestión de redes se haya vuelto más innovador gracias a su capacidad de programación y su facilidad de configuración.

Por otro lado, en la administración y gestión de redes la tendencia es confiar cada vez más en el software, lo que acelerará la innovación de las redes.

Pero el hecho de que esta tecnología sea relativamente nueva, representa un freno para algunas empresas. En general, desde una perspectiva empresarial, los beneficios aún están por llegar. Por ejemplo, los grandes proveedores de servicios, con redes que soportan un gran estrés por los altos volúmenes de tráfico, podrían obtener beneficios de esta tecnología en un menor plazo. En cambio, la mayoría de las empresas aún mantendrán sus ya probadas infraestructuras existentes, que a día de hoy les ofrecen una mayor confiabilidad que los nuevos productos SDN.

Antes de que la mayoría de las empresas decidan migrar a SDN, necesitarán algunas garantías sobre temas de seguridad, confiabilidad e interoperabilidad con sus soluciones. La mayoría esperarán a que SDN evolucione y que cuajen los estándares de interoperabilidad.

Esto no quiere decir que los administradores de TI puedan olvidarse de esta tecnología, ya que el futuro de SDN parece venir cada vez más rápido. SDN está en el camino de convertirse en el nuevo "estándar" para redes en un futuro no muy lejano.

En que consiste [1], [2], [3], [4], [5], [8], [9], [10], [12]

Una vez visto cómo y porqué aparecieron las redes definidas por software, es hora de pasar a describir en qué consisten, es decir, como funcionan, que aportan, sus ventajas,…

Las redes definidas por software (SDN), son un conjunto de técnicas cuyo objetivo es facilitar a los administradores de red la gestión de servicios de red a través de la abstracción de funcionalidades de bajo nivel. Estas abstracciones de red se pueden utilizar fácilmente mediante unas interfaces API.
Se podría decir que la finalidad de SND es facilitar la implementación y posterior implantación de servicios de red de una manera sencilla, dinámica y escalable, evitando al administrador de red gestionar dichos servicios a bajo nivel.
Cualquier equipo de red puede realizar dos funciones básicas:

- Permitir el intercambio de información y verificar el comportamiento del tráfico de la red. Esto se realiza en la capa de control.
- Reenviar o descartar paquetes en función del estado de la red, que le es comunicado por la capa de control. Esto se realiza en la capa de datos.

Como ya se ha dicho, SDN abstrae las funcionalidades de bajo nivel. Para conseguir esta abstracción separa las dos capas explicadas anteriormente:

- Toma de decisiones (plano de control), que se correspondería con el software.
- Sistema encargado de encaminar el tráfico a su destino (plano de datos), que se correspondería con el hardware.

La idea principal reside en el hecho de gestionar las redes separando el plano de control del plano de datos. Esta separación entre el software y el hardware hace que este último se pueda desprender de parte de sus procesos, consiguiendo así ser más eficiente en las tareas de conmutación.
Pero al haber hecho esta separación aparece la necesidad de utilizar algún método o mecanismo para comunicar estos dos planos, uno de esos mecanismos es el protocolo OpenFlow (este protocolo se verá con más detalle posteriormente).
Es decir, SDN es una manera de enfrentarse a la creación de redes en la cual el control se desprende del hardware y se le da a una aplicación de software, llamada controlador.

Actualmente las redes convencionales se basan en el uso de routers en los cuales el plano de control (protocolos de routing, listas de acceso, políticas…) y el plano de datos (switching, routing) están unidos. Lo que hace que el operador se vea obligado a adaptarse a las características funcionales de cada fabricante de hardware.
Además, cuando un paquete llega a un switch, las reglas integradas en el firmware propietario del switch le dicen a este donde debe enviar el paquete. Por lo tanto, el switch envía cada paquete al mismo destino y por la misma trayectoria, es decir, trata todos los paquetes de la misma manera.
En cambio, las redes definidas por software se centran en el plano de control y lo utilizan para establecer la lógica de funcionamiento de la red formada por switches/routes. Desde la parte central (SDN controller) se aplica la lógica de switching/routing (flow tables) a los equipos de red por medio de protocolos como OpenFlow. Las operaciones de switching/routing se realizan basándose en las reglas almacenadas en las flow tables de los switches/routes.

Esto permite a un administrador de red modificar el tráfico desde una consola de control centralizada sin tener que tocar switches individuales. El administrador puede cambiar cualquier regla de los switches de red cuando sea necesario, es decir, le permite tener más control que nunca sobre el flujo del tráfico de red.
Actualmente, la especificación más extendida para crear una red definida por software es un estándar abierto llamado OpenFlow, que permite a los administradores de red controlar tablas de enrutamiento de forma remota.

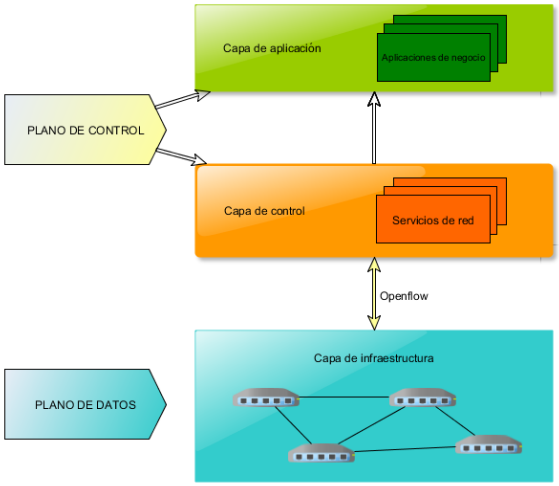


Figura 2.1 Arquitectura simplificada de SDN que utiliza el protocolo OpenFlow

Los proveedores de redes SDN ofrecen una amplia variedad de arquitecturas, centradas en el objetivo de separar el plano de control del plano de datos. Además de estos dos planos o capas, estas arquitecturas también están formadas por al menos:

- **Un controlador SDN**.

  Es el punto de control estratégico que envía información a través del southbound API a los switches/routers de la capa inferior y a través del northbound API a las aplicaciones y la lógica de negocio de la capa superior.

  Estos controladores se basan normalmente en un conjunto de módulos que se pueden conectar y desconectar libre y fácilmente. Estos módulos se encargan de diferentes tareas de red, como realizar un inventario de todos los aparatos de red disponibles, guardar sus capacidades, agrupar estadísticas de red, etc.
- **Una API "hacia el sur" (southbound API).**

  Su finalidad es facilitar el control en la red, permitiendo al controlador realizar cambios en tiempo real de acuerdo a las necesidades y demandas que se presentan.
- **Una API "hacia el norte" (northbound API).**

  Estas interfaces son las más críticas en las redes SDN, debido a que soportan una gran variedad de servicios y aplicaciones por encima, con la posibilidad de que algunas de ellas no funcionen correctamente. Estas interfaces son el componente más indeterminado de todo el entorno SDN, ya que existe una gran variedad de posibles interfaces de este tipo para controlar los diferentes tipos de aplicaciones a través del controlador SDN.

Como se puede observar en la figura 2.2, estas interfaces permiten conectar el controlador SDN, por un lado, con los equipos de red de la capa inferior (southbound), y por otro, con los servicios y aplicaciones de la capa superior (northbound).

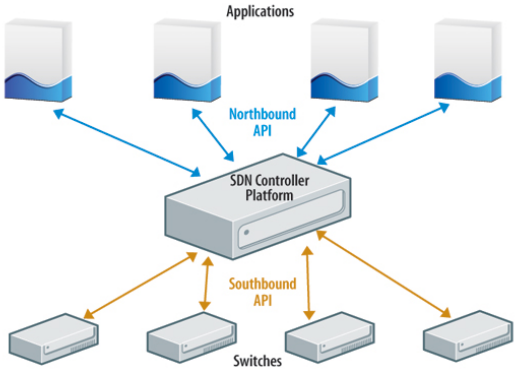


Figura 2.2 Controlador SDN que utiliza las APIs

Una vez explicado que son y cómo funcionan las redes SDN, se puede ver que son muchos sus beneficios. A continuación se exponen de forma resumida algunas de las principales ventajas que pueden aportar a las organizaciones:

- **Agilidad y flexibilidad.** SDN permite a las organizaciones desplegar servicios, aplicaciones e infraestructuras de forma rápida permitiendo así alcanzar los objetivos marcados en el menor tiempo posible.
- **Permite innovación.** Los entornos SDN permiten crear nuevos tipos de aplicaciones y modelos de negocio por parte de las empresas.
- **Reduce el gasto operacional.** La tecnología SDN permite el control de elementos de red mediante el uso de programas, como switches/routers, que cada vez son más programables. Esto hace que sea más sencillo configurar y gestionar las redes. En consecuencia, esto permite una reducción del tiempo de gestión por parte de los administradores y reduce considerablemente la probabilidad de un error humano.
- **Reduce el gasto en hardware.** Los entornos SDN permiten reutilizar el hardware existente alargando así su vida útil. Esto limita la necesidad de renovar el hardware existente y de tener que invertir en hardware nuevo.
  Además, permite a los administradores de redes disponer de hardware de diferentes fabricantes.

Pero a pesar de las ventajas, beneficios y mejoras de SDN explicados anteriormente, siempre existen algunos riesgos o puntos débiles.
Observando y analizando la arquitectura de SDN, rápidamente se puede observar cuál es su principal punto débil o talón de Aquiles, el controlador.
Esto se debe a que el controlador, en las redes SDN, es el centro de control y de decisión de la red. Por lo tanto hay que protegerlo y asegurarlo, ya que un problema en el controlador afectará a toda la red.
Los problemas más importantes en los entornos SDN pueden ser provocados por:

- **Escalabilidad.** Un gran aumento de la carga de trabajo del controlador SDN podría convertirlo en un “cuello de botella” o incluso llegar a “saturarse”. Y al ser el elemento único de control, un problema en él puede afectar a toda la red.

  Por ejemplo, si se produce un gran aumento del número de clientes y aplicaciones y no se aumentan los recursos del controlador.
- **Seguridad.** Los ataques al controlador podrían afectar al funcionamiento de toda la red. Es por esto por lo que debe estar especialmente protegido con todos los mecanismos de seguridad necesarios para asegurar su disponibilidad. Para su protección se pueden utilizar las herramientas de seguridad que se aplican en las redes convencionales, que se pueden adaptar a las redes definidas por software. No obstante, las organizaciones están estudiando mecanismos de seguridad específicos para SDN que se adapten mejor a esta

tecnología. Centrándose especialmente en el controlador, que es el punto central de estas arquitecturas y ha de ser el elemento más securizado.

Por ejemplo, si se lanzara un ataque de DoS contra el controlador y este cayera, toda la red caería también.

Resumiendo, SDN se caracteriza por separar la capa de control de la de datos, permitir un control centralizado y ofrecer la capacidad de programar y automatizar el comportamiento de la red. De esta forma se consigue optimizar el uso de los recursos de red, aumentar su agilidad y flexibilidad, y permitir la innovación y el dinamismo de las redes.
Además, desde el punto de vista de los costes, SDN permite reducir los gastos económicos y operativos. Esto lo consigue proporcionando unas abstracciones de red fáciles de utilizar mediante interfaces API, lo que permite a más personas, e incluso a sistemas automatizados, realizar la provisión y configuración de la red. Y todo esto reduciendo los tiempos de respuesta a minutos o segundos y reduciendo considerablemente la probabilidad de un error humano.

2.3 Estudio de la tecnología NFV.

Origen y evolución. En que consiste.

Origen y evolución [1], [6], [7]
NFV son las siglas de Network Functions Virtualization (Virtualización de las Funciones de la Red, en español).
Los orígenes de NFV se sitúan en Octubre de 2012, cuando un grupo de especificadores de la industria (Network Functions Virtualization) publicó un "white paper" en una conferencia en Darmstadt sobre Software-defined networking y OpenFlow. Este grupo estaba compuesto por proveedores de servicios de red como AT & T, BT, China Mobile, Deutsche Telekom, entre otros.

Tradicionalmente la industria de las telecomunicaciones, a la hora de desarrollar sus productos, ha seguido estándares muy estrictos de calidad, estabilidad y cumplimiento de protocolos. Pero este modelo, que funcionó muy bien en el pasado, se ha quedado obsoleto. Los principales motivos han sido que este modelo ha conducido inevitablemente a un ritmo de desarrollo muy lento que provoca ciclos de desarrollo de producto muy largos. Además, también ha provocado una gran dependencia de hardware propietario muy especializado.

Actualmente, NFV, está en fase de desarrollo por el Grupo de Especificaciones de Industria del Instituto Europeo de Normas de Telecomunicaciones (ETSI) y desde la publicación del documento inicial, se han creado otros materiales que tratan el tema con mayor profundidad. Entre otras cosas incluyen una terminología estándar y casos de uso para NFV con el objetivo que sirvan como referencia para los fabricantes y los operadores que decidan realizar productos o soluciones relacionadas con esta tecnología.
El gran aumento de la competencia en los servicios de comunicaciones ofrecidos por algunas de las organizaciones que operan a gran escala en internet (como Google Talk, Skype, Netflix,…), ha estimulado a los proveedores de servicios para buscar la manera de alterar el statu quo existente.
La introducción de NFV será un proceso gradual y en varias etapas. La interconexión con las redes tradicionales será un elemento crítico para garantizar servicios ininterrumpidos durante la evolución hacia una infraestructura basada totalmente en NFV. No obstante, es poco probable que los hosts virtuales desplacen totalmente a los equipos de red, pero la parte de alto valor de los servicios de red podría convertirse en

una serie de componentes alojados en la nube, que interactúan.
Algunos ejemplos de VNFs que se están utilizando son los aceleradores de WAN, firewalls, seguridad, balanceadores, etc., es decir, todas las aplicaciones que hasta ahora se realizaban por medio de appliances. Además, se pueden añadir funciones típicas de routing como IPsec, túneles, routing dinámico, etc.

En que consiste [1], [2], [3], [6], [7], [8], [9], [10], [11]
NFV es un concepto de arquitectura de red que propone utilizar técnicas relacionadas con la virtualización de TI, con la finalidad de virtualizar diversos tipos de funciones de nodos de la red, con el objetivo de poder conectarlos para crear servicios de comunicación.
Como ya se ha dicho, NFV se basa en tecnologías relacionadas con la virtualización de TI, concretamente utiliza Funciones de Red Virtualizadas (VNF del inglés Virtualized Network Function).
VNF consiste en una o más máquinas virtuales que ejecutan diferentes programas y procesos, sobre servidores estándar de alta capacidad, switches o infraestructuras cloud computing, en lugar de funcionar sobre dispositivos de red especializados.
Con un ejemplo se verá más claro, por ejemplo, se podría desplegar un firewall para proteger una red sin el coste y la complejidad que supondría desplegar un dispositivo físico.
El conjunto de servidores sobre los que corren las VNFs, constituyen la red NFVI (del inglés NFV Infraestructure). Estos servidores pueden estar situados en cualquier punto de la red del operador.
Dicho de una manera algo más clara, NFV permite implementar funciones de red mediante software, en lugar de hardware, es decir, virtualiza funciones de red sobre servidores y/o switches.
De esta forma, si se quiere implementar, por ejemplo, una función de cifrado de red, en lugar de hacerlo con un nuevo dispositivo hardware, se puede hacer con un software de cifrado en el servidor o en el switch, desde cualquier lugar dentro de la red. Esto hace que NFV permita reducir la dependencia de dispositivos hardware, aumentando la escalabilidad y personalización de la red de forma que sea más fácil mejorarla. Además, al evitar el hardware consigue reducir tanto el espacio que éste ocupa como su consumo energético, además de reducir los costes de mantenimiento.

Pero una VNF por sí sola no proporciona automáticamente un producto o servicio utilizable por los clientes de un proveedor. Para poder construir servicios más complejos, se utiliza el concepto de “encadenamiento de servicios” o “Cadenas de Funciones de Servicios (SFC)”, donde se utilizan múltiples VNFs para ofrecer un servicio. Por lo tanto, un proveedor de servicios que sigua el diseño NFV implementará una o más funciones de red virtualizadas (VNFs). Dicho de otro modo, una cadena de servicios es una secuencia de funciones VNF que se añaden en el recorrido de la conexión de datos, entre el punto de origen y el punto de destino del tráfico de red.
Esto permite a los proveedores de servicios crear y comercializar paquetes de servicios de valor añadido añadiéndolos dinámicamente a la secuencia de las conexiones de datos del cliente.
Generalmente, cada instancia de estos componentes de software definidos por el proveedor se mapean en una máquina virtual.

Por ejemplo, en la figura 2.3 se puede observar como la cadena de servicios de color azul incluye un firewall y un sistema de optimización de vídeo. Mientras que la cadena de color rojo también incluye el firewall, pero no la optimización de vídeo y añade las funciones de antivirus y control paterno.

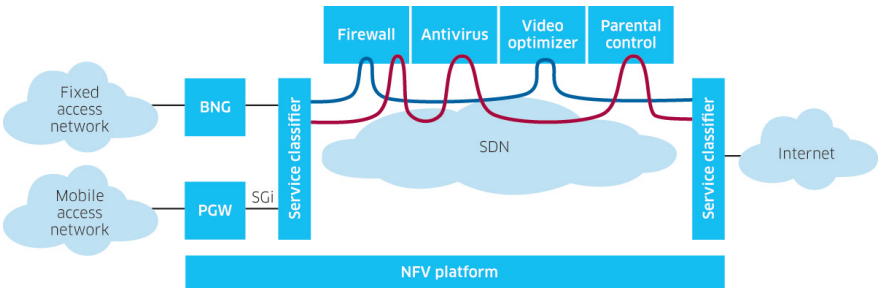


Figura 2.3 Ejemplo de Cadenas de Funciones de Servicios

Virtualizar las funciones de red, permite a los proveedores flexibilizar la gestión del flujo de tráfico y el uso eficiente de los recursos dentro de sus complejas redes. Esto se consigue gracias a que NFV transforma toda la arquitectura de red de los operadores al alojar sus funciones de red en máquinas virtuales (VMs), en lugar de dispositivos hardware como hasta ahora. Esto hace que el traslado de los servicios de red al Cloud se pueda realizar de forma ágil, fiable y eficiente. Sin menoscabo de que aquellos servicios que necesitan un hardware dedicado se puedan trasladar a servidores estándar.
Todo esto es posible gracias a la infraestructura de red flexible de NFV, en la que las funciones de red están bajo el control de un hipervisor.

Un elemento muy importante en las soluciones NFV es el vSwitch. El vSwitch es un componente clave de las plataformas NFV ya que es el responsable de proporcionar conectividad tanto entre máquinas virtuales como entre las máquinas virtuales y la red exterior. Su rendimiento determina tanto el ancho de banda de los VNFs como la relación coste-eficiencia de las soluciones de NFV.

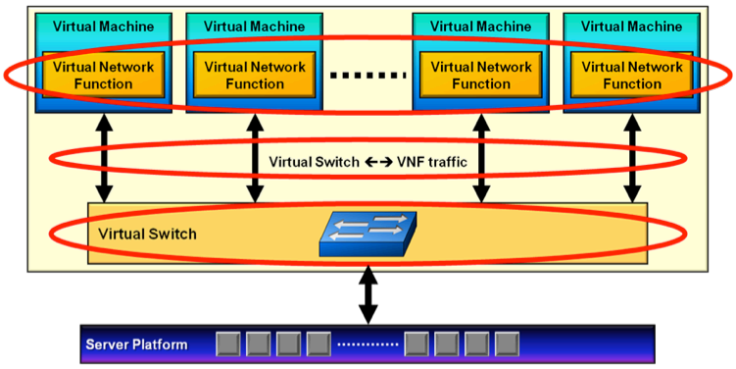


Figura 2.4 Ejemplo de arquitectura NFV con vSwitches

Otro aspecto importante de la implementación de NFV es el proceso de orquestación. Para construir servicios altamente confiables y escalables, NFV requiere que la red sea capaz de crear instancias de casos VNF, monitorizarlas, repararlas, y (lo más importante para una empresa de servicios) facturar por los servicios ofrecidos. Para conseguir estos objetivos, estos atributos se asignan a una capa de orquestación con la finalidad de proporcionar una alta disponibilidad y seguridad, además de mantener los costes de operación y mantenimiento bajos. Además, mover las funciones de red de dispositivos de red dedicados (routers, switches, firewalls,…) hacia servidores básicos ayuda a los proveedores a evitar una sobrecompra de equipos y la subutilización de recursos.
Es importante mencionar que la capa de orquestación debe ser capaz de gestionar VNFs independientemente de la tecnología subyacente dentro de la VNF. Por ejemplo, una capa de orquestación debe ser capaz de gestionar un VNF SBC del proveedor X

corriendo en VMware vSphere tan bien como un VNF IMS del proveedor Y corriendo en KVM.

A grandes rasgos, la arquitectura de NFV se especifica mediante una serie de bloques de funciones comunes que abarcan la totalidad del hardware y software que se utiliza para construir la red. Tal y como se puede ver en la figura 2.5.

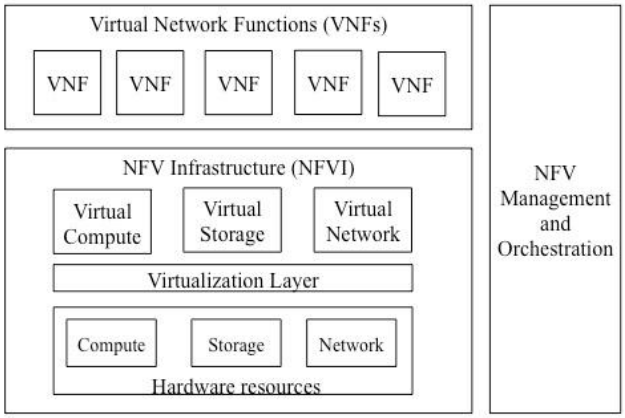


Figura 2.5 Marco de referencia NFV

Estos bloques se podrían dividir en tres y son:

- **Funciones de redes virtuales (VNF)**. Estas virtualizan un "elemento de red" o una función de elemento de red, como por ejemplo un router, un switch o una estación base. Estas funciones están gestionadas por el correspondiente controlador (VNF manager) que utiliza interfaces abiertas para poder comunicarse con los elementos de red o bien directamente, o bien a través de los sistemas de gestión de elementos de red (EMS) que pueden ser propietarios de cada fabricante.
- **Infraestructura NFV (NFVI)**. Consiste en el hardware y software que sirve para construir los VNF, por ejemplo los servidores x86, en forma de recursos físicos de hardware.
- **Virtualizador o hipervisor.** Desacopla el software del hardware mediante la abstracción de recursos como la memoria virtual y la CPU. Este virtualizador está a su vez gestionado por su gestor de infraestructura virtual (VIM) y el orquestador. Entre ambos generan, mantienen y permiten la provisión de servicios entre diferentes VNF.

Y todo esto permite a NFV, por ejemplo, realizar una provisión de servicios bajo demanda y su extensión extremo a extremo a lo largo de la red, incluso aunque esta sea de diferentes operadores y utilice diferentes tecnologías. Además de permitir la adaptación e integración con los elementos de cálculo (servidores) y de almacenamiento para poder construir soluciones adaptables a la carga de trabajo o para crear arquitecturas distribuidas, o que sigan prestando servicio en caso de averías, de forma transparente para el usuario.

Una vez explicado que es y cómo funciona NFV, se puede ver que son muchos sus beneficios. A continuación se procederá a exponer de forma resumida algunas de las principales ventajas que pueden aportar a las organizaciones:

- El "tiempo de lanzamiento al mercado" (time to market) se reduce. Esto es posible gracias a que al no ser imprescindible un hardware específico (es una cuestión de software), el tiempo necesario para tener operativa una nueva funcionalidad de red se reduce considerablemente. Es decir, permite adaptarse

dinámicamente a los cambios y desplegar soluciones a medida de cada cliente.

- Es una solución muy dinámica ya que se pueden aumentar o reducir las capacidades de cualquier VNF de acuerdo a las necesidades específicas. Permite asignar los recursos necesarios en cada momento en función de las demandas puntuales de los clientes.
- A la hora de crecer, simplemente se provisiona más infraestructura virtualizable y la capacidad, en términos generales, crece considerablemente. Pero no crece sólo la capacidad operativa, sino también la capacidad de "supervivencia" porque se distribuyen las funcionalidades en un mayor número de dispositivos físicos que se interrelacionan.
- Contribuye al aumento de la competitividad de los operadores gracias a la reducción del CAPEX y OPEX. El uso de plataformas hardware de propósito general en vez de máquinas dedicadas, hace que se reduzcan los gastos en inversión y explotación (CAPEX). Además, se produce una bajada del consumo energético y logístico, y una mejora y simplificación de la automatización de la red (OPEX).
- Reduce la "osificación" de la red al permitir la innovación y su implantación de forma rápida.
- Las operaciones de red se simplifican al poder realizarse de forma centralizada permitiendo así una automatización y gestión del flujo del tráfico más eficaz.
- Ofrece una gran disponibilidad ya que en caso de fallo de un elemento, la funcionalidad que este ofrecía pasa a ofrecerla otro elemento rápidamente.
- Se produce un aumento del "retorno de la inversión" (RoI). Las implementaciones NFV, gracias a muchas de las ventajas expuestas en los puntos anteriores, hacen que se alcance el retorno de inversión mucho antes que con los despliegues tradicionales.

Pero a pesar de las ventajas, beneficios y mejoras de VNF explicadas anteriormente, siempre existen algunos riesgos o puntos débiles. En este caso, hay que tener muy en cuenta la seguridad.
Lo más habitual es que un operador tenga sus funciones de red virtualizadas en un cloud. Por lo tanto, este operador estará expuesto a una mayor o menor amenaza en función del cloud que utilice. En general los clouds son muy seguros, pero ninguna tecnología es infalible y por eso es imprescindible que el proveedor de la infraestructura cloud disponga de los mecanismos adecuados para garantizar la seguridad del sistema.

Resumiendo, en la actualidad muchos servicios se soportan en sistemas independientes de servicios. Cada servicio está compuesto por su propio hardware, software y equipo de operaciones. Con NFV, los proveedores de servicios pueden desplegar un modelo más horizontal en el que la plataforma NFV (con sus recursos de red, almacenamiento y computación) es una capa común que no es necesario duplicar para cada servicio. Esto cambia las funciones y responsabilidades de los equipos de operaciones.
Para ello, NFV permite que muchos de los equipos que se utilizan actualmente, como por ejemplo: CDNs, routers de mensajería, session border controllers, DPI, SGSN/GGSN, nodos de red fijos y móviles y BRAS, entre otros, sean reemplazados por servidores de alta densidad, switches y servidores de almacenaje. Con este cambio en el tipo de infraestructura, el software pasa a jugar un papel mucho más relevante, porque además éste puede moverse por la red (hardware) de forma dinámica y aprovechar así mejor sus recursos.
NFV permite la instalación de estas funciones de red en dispositivos virtualizados. Gracias a esto, para conseguir la portabilidad en las aplicaciones de software, sólo es necesario implementar hipervisores que proporcionen la suficiente abstracción del hardware.

2.4 Fabricantes y proveedores de servicios implicados.

Como ya se ha visto en los apartados anteriores, a pesar de que las tecnologías SDN y NFV son relativamente nuevas, están generando muchas expectativas ante las cuales el mercado del networking se está moviendo. Los grandes proveedores (y algunos no tan grandes) ya se han movido y han lanzado algunas soluciones. En cambio otros han optado por la opción de establecer alianzas o realizar compras. Todo ello con el objetivo de conseguir hacerse un hueco dentro de este mercado emergente.

No hay que olvidar que estas tecnologías son todavía algo inmaduras, y por lo tanto les hacen falta desarrollos y pruebas antes de que las empresas, especialmente las PYMES, adopten tecnologías SDN o NFV de forma generalizada. Saber cuándo y de qué manera adoptar estas nuevas tecnologías es uno de los grandes retos del nuevo y complejo entorno empresarial. Por ello, disponer de un socio tecnológico experimentado y con visión de futuro ayudará a las PYMES a lanzarse a la piscina tecnológica sin ahogarse. Es decir, puede guiarlas a implementar las tecnologías adecuadas a cada empresa midiendo los tiempos y ajustándose a las necesidades particulares de cada una. Es por esto por lo que de momento sólo las grandes empresas se están atreviendo a meterse en el sector de las tecnologías SDN y NFV.

Pero en este sector hay que tener cuidado con esperar demasiado, no hay que dormirse en los laureles porque todo va muy rápido. Según la consultora IDC, se prevé que el mercado de SDN alcance una cifra de negocio de 3.700 millones de dólares en 2016, frente a los 252 millones de 2012, o los 10 millones de 2007 [5], [12].

A continuación se exponen algunos ejemplos de empresas implicadas con estas tecnologías y algunos de los productos y soluciones relacionados con SDN/NFV que están ofreciendo. Esta relación de empresas intenta ser una selección representativa de empresas implicadas con SDN/NFV ya que no se pueden exponer todas. Del mismo modo, los productos o soluciones que se explican son también una selección de todas las existentes. Muchas empresas disponen de un gran portfolio de productos y soluciones SDN/NFV, aquí sólo se ha explicado uno de cada empresa a modo de ejemplo.

Cisco [12], [15]

Cisco se lo ha tomado en serio y ya ofrece tecnologías SDN en su solución Cisco ONE (del inglés Open Network Environment), que se presenta como una red inteligente caracterizada por su programabilidad y por ofrecer una amplia gama de plataformas API, agentes, controladores y tecnología de red superpuesta en un kit.

Cisco ONE Software es una manera flexible y óptima de comprar software para los centros de datos, WAN y dominios de acceso. En cada etapa del ciclo de vida del producto, el software Cisco ONE facilita y ayuda en la compra, gestión y mejora de la red y la infraestructura de software.

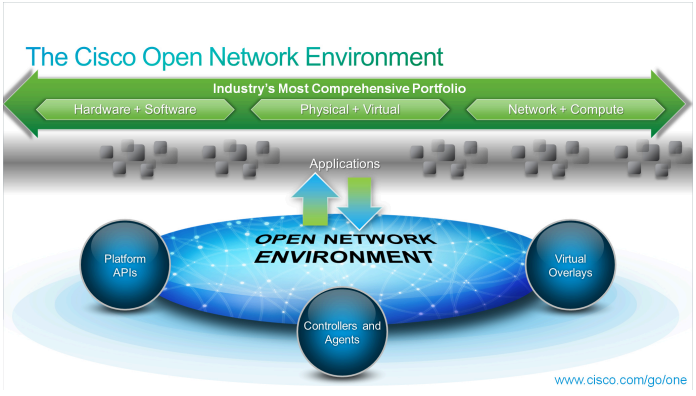


Figura 2.6 Esquema de entorno Cisco Open Network Environment (ONE)

HP [12], [16], [24]

HP dispone de productos para las capas de infraestructura, de control y de aplicación. Su cartera incluye más de 50 switches compatibles con SDN y con más de 30 millones de puertos instalados en todo el mundo. Por otro lado, HP también ofrece un "mercado" para aplicaciones SDN (SDN App Store) y una plataforma para compartir innovaciones.

HP apuesta por la virtualización de la red con Virtual Application Networks (VAN), que utiliza elementos de tecnologías SDN, incluyendo OpenFlow, que está disponible en diferentes switches HP Networking.
Concretamente, el controlador SDN de HP, Virtual Application Networks (VAN), proporciona un punto de control unificado en una red compatible con SDN, simplificando la gestión, el aprovisionamiento y la orquestación. Es decir, el controlador permite el control centralizado de la red y la automatización de las funciones de red, incluyendo funciones tales como la detección de la topología de red. Además, dentro de un entorno HP SDN, el controlador VAN ofrece integración entre la red y el sistema de negocios de la organización.
El controlador VAN también puede agruparse, lo que permite a un controlador hacerse cargo de las funciones de otro si uno falla. En lo que respecta a la seguridad, el controlador utiliza métodos de autenticación y autorización. A su vez, las aplicaciones SDN pueden interactuar con el controlador, mientras que las aplicaciones no autorizadas no pueden tener acceso a la red. Las conexiones hacia el sur entre los switches OpenFlow y el controlador de HP también están aseguradas y encriptadas.
HP SDN ofrece una red programable que está alineada con las aplicaciones comerciales y basada en estándares abiertos.

Por otro lado, HP también está trabajando con Verizon e Intel para desarrollar una app con la finalidad de proporcionar ancho de banda WAN utilizando el controlador VAN.

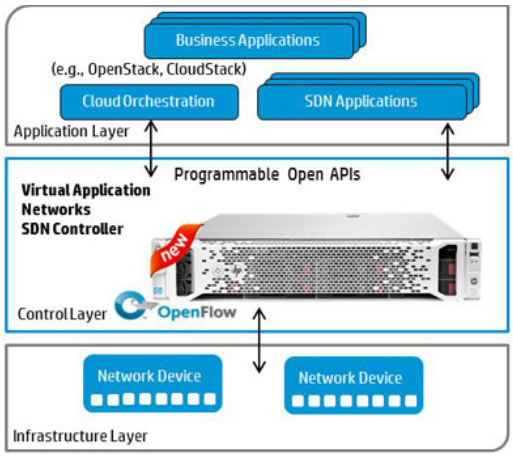


Figura 2.7 Esquema de entorno HP Virtual Application Networks (VAN)

Microsoft [19]

Por su parte Microsoft, se ha visto obligada a utilizar SDN para poder implementar Azure cloud. Pero en su caso, al igual que Google, han decidido desarrollar su propio software SDN.

La nube de Microsoft, a través del cual se entregan los productos de software de la compañía, cuenta con 22 regiones hyper–scale en todo el mundo. El almacenamiento y el uso computacional de Azure se duplica cada seis meses, y las líneas de Azure aumentan en 90.000 nuevos suscriptores al mes.
Con el fin de mantenerse al día con ese tipo de crecimiento, Azure necesita un diseño

virtualizado, particionado y escalable horizontalmente, entregado a través de software. Para conseguir estos objetivos, Microsoft escribió todo el código del software para SDN de Azure.
Aunque SDN y código abierto van mano a mano, no hay contenido de software de código abierto en el Azure SDN. Esto se debe a que la funcionalidad requerida para Azure no se ofrece a través de las comunidades de código abierto.

IBM [12], [17]
IBM es miembro fundador de tecnologías abiertas como OpenDaylight, diseñada para acelerar la adopción de las redes SDN; OpenStack, un sistema operativo cloud diseñado para controlar y organizar grandes reservas de recursos a lo largo de un centro de datos; y la fundación Cloud Foundry Foundation, que ayuda a impulsar los estándares globales de PaaS.
IBM considera que SDN juega un papel fundamental permitiendo a las organizaciones ofrecer servicios de TI de la manera más eficiente posible.
Por eso, basándose en su arquitectura DOVE (Distributed Overlay Virtual Ethernet), ofrece SDN para Virtual Environments (VE), un controlador de red programable y switches OpenFlow.
Concretamente, IBM SDN para entornos virtuales (VE) proporciona una arquitectura SDN común que aporta los beneficios de la virtualización de red a entornos de nube. Los clientes pueden desplegar aplicaciones y entornos de carga de trabajo utilizando redes integradas superpuestas a través de hipervisores.

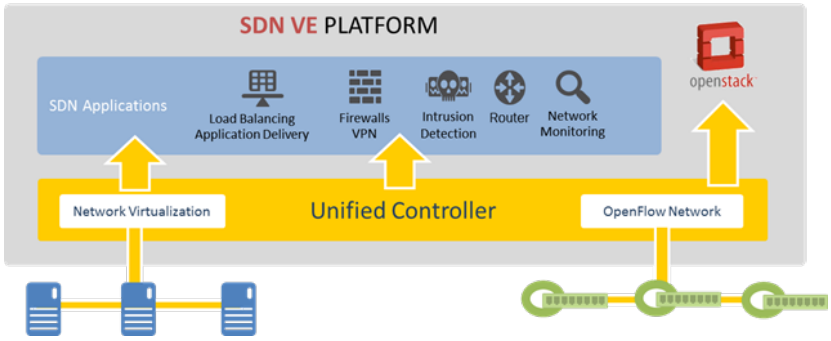


Figura 2.8 Esquema de plataforma IBM SDN Virtual Environments (VE)

Google [14], [18]
[14] En 2012 Google estaba tan impresionado con los beneficios de SDN que decidió extender su actual red de “inter-data center network” internacionales y construir otras redes nuevas con las mismas capacidades, declaró Jim Wanderer, su director de servicios web.
Google ya había hablado públicamente acerca del uso del protocolo OpenFlow SDN en su red interna que conectó los centros de datos en América del Norte, Europa y Asia/Pacífico y los beneficios económicos asociados que aportaba.
Jim Wanderer justificó esta decisión diciendo que la “inter-data center network” era necesaria y podría haber sido construida utilizando tecnología heredada. Pero, según él, el problema era que esos sistemas "*no se comportaban de una manera determinista, eran difíciles de configurar y operar a escala, eran propensos a errores y además eran caros*".
Wanderer y su equipo querían optimizar el enrutamiento y centralizar la ingeniería de tráfico. Así que se desplegó hardware básico, con OpenFlow como protocolo SDN al lado de la suite de software de enrutamiento Quagga, que era compatible con la red existente de Google y conocida por el equipo de operaciones.
El equipo utilizó varios controladores OpenFlow (para baja latencia y conectividad confiable con los dispositivos conectados) y la ingeniería de tráfico centralizada, y

entró en producción a principios de 2011.
Wanderer declaró que, "*en general, SDN ha trabajado muy bien, va en todos nuestros proyectos. Han habido grandes dividendos desde el uso de dispositivos de alta computación en el plano de control. No podríamos haber logrado los resultados que hemos obtenido sin SDN. Podríamos haber construido alguna otra cosa para hacer esto, pero no habría sido tan eficaz.*"

[18] Más recientemente, (en junio de este año) en una presentación de apertura en el ONS (del inglés Open Network Summit), Amin Vahdat, responsable técnico de redes de Google, describió la arquitectura de red de centro de datos de la compañía.
Resumiendo la presentación de Vahdat, este dijo que la red se organiza en torno a una topología de Clos en una colección de pequeños y baratos switches, que se agrupan dentro de un switch lógico mucho más grande.
Google utiliza un software de control de pila centralizado, escrito internamente, para gestionar miles de switches dentro del centro de datos y tratarlos como una maya de gran tamaño. Y en la construcción de su propio software y hardware, Google se basa menos en protocolos estándar de Internet y mucho más en los protocolos personalizados a la medida de sus centros de datos. Google ha estado utilizando y beneficiándose de su propio SDN durante los últimos 10 años, consiguiendo durante este periodo de tiempo aumentar la capacidad de una sola red de centro de datos en más de 100 veces.

Dell [13]
Dell ofrece un enfoque completo de SDN. Diseña sus soluciones proporcionando capacidades de vanguardia mediante la adopción de las mejores innovaciones arquitectónicas de las últimas aplicaciones SDN y su entrega a través de plataformas y tecnologías seguras.
Su solución Active Fabric está diseñada para SDN, con soporte para Network Virtualization Overlays (NVOs) utilizando los principales hipervisores de Microsoft, VMware y OpenStack. También soportan los controladores basados en OpenFlow de los principales proveedores. Su Active Fabric también soporta interfaces de programación incluyendo Telnet/CLI, TCL, REST, SNMP, Perl y Python scripting.

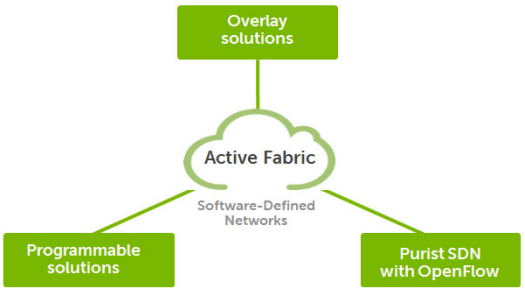


Figura 2.9 Esquema de la solución SDN Active Fabric de Dell

VMware [24]
VMware ofrece el controlador NSX. Este controlador se considera un sistema de gestión de estado distribuido que permite controlar las redes virtuales y superponer los túneles de transporte. Este controlador se convierte en el punto de control central para todos los switches lógicos dentro de una red. El controlador mantiene la información de las máquinas virtuales, hosts, switches lógicos y VXLANs, mientras usa APIs en dirección norte (northbound) para hablar con las aplicaciones.
Cuando se trabaja con el controlador, las aplicaciones comunican lo que necesitan, y el controlador programa todos los switches virtuales bajo el control NSX en dirección

sur (southbound) para cumplir con esos requisitos. El controlador permite su ejecución de dos maneras diferentes dentro de NSX:

- Como un cluster apartado de máquinas virtuales en un entorno vSphere.
- En los aparatos físicos para aquellos con hipervisores mixtos.

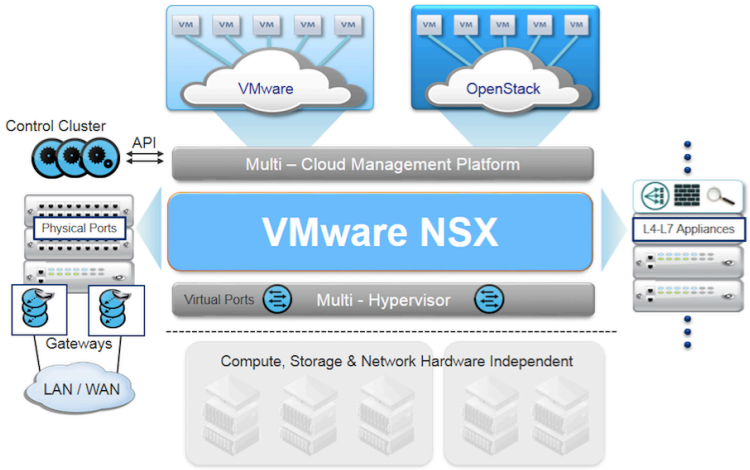


Figura 2.10 Esquema de la solución propuesta por VMware (NSX)

NEC [24]

NEC ofrece el controlador ProgrammableFlow PF6800. Este controlador se sitúa en el centro del tejido de red basado en OpenFlow ProgrammableFlow de NEC. Proporciona un punto de control para las redes físicas y de gestión tanto para las redes virtuales como para las físicas. El controlador se considera programable y estandarizado ya que se integra tanto con OpenStack, como con Microsoft System Center Virtual Machine Manager para una gestión de red y orquestación añadidas. El controlador también incluye la tecnología de red de inquilino virtual de NEC, que permite redes aisladas, con varios inquilinos.

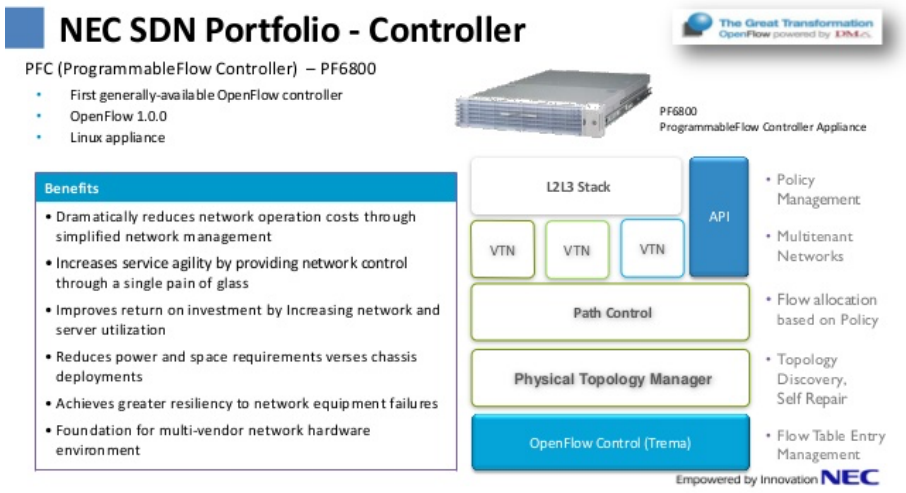


Figura 2.11 Características del ProgrammableFlow PF6800 de NEC

Nuage Networks [24]

Nuage Networks es una empresa de la multinacional francesa Alcatel-Lucent dedicada a proporcionar las herramientas necesarias para permitir una infraestructura de redes en Cloud.

Esta empresa ofrece el Controlador de Servicios Virtualizados (VSC). Este controlador

permite la visión completa de una red por inquilino y las topologías de servicio, mientras externaliza plantillas de servicios de la red definidos a través del Directorio de Servicios Virtualizados (VSD) de Nuage Networks. El directorio de servicios es un motor de políticas que utiliza el análisis y las reglas de red para autorizar permisos basados en roles. El VSC envía mensajes usando esas reglas a la plataforma de conmutación y ruteo virtual de Nuage. La plataforma detecta tanto la creación como la eliminación de una máquina virtual y luego le pregunta al controlador SDN si hay alguna política instalada para ese inquilino. Si existe una regla, la conectividad de red se establece inmediatamente.

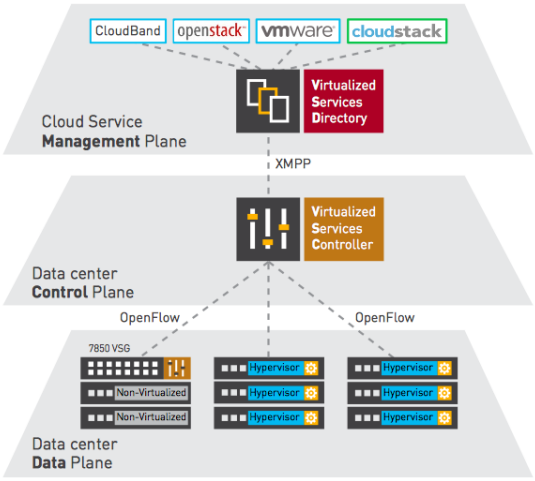


Figura 2.12 Esquema de la arquitectura propuesta por Nuage Networks

Huawei [29]

Huawei es otra empresa que apuesta fuertemente por las tecnologías SDN y NFV. Prueba de ello es que se clasificó como principal proveedor para soluciones Software-Defined Networking (SDN) y Network Functions Virtualization (NFV) por segundo año consecutivo, según una encuesta de 2014 realizada por la firma de análisis Current Analysis a operadores de telecomunicaciones líderes.

La encuesta "SDN and NFV: Evolving Deployment Requirements and How to Benefit" se centró en evoluciones del mercado SDN y NFV, preferencias de compra, capacidades de vendedores y posicionamiento.

Según Ken Wang, director general de Marketing Global y Ventas de Soluciones de Huawei, Huawei ha estado promoviendo la comercialización de SDN/NFV desde la introducción de la estrategia SoftCOM en 2012. Además, hasta la fecha, se han asociado con más de 20 compañías de telecomunicaciones para realizar más de 60 proyectos de innovación SDN/NFV conjuntos, consiguiendo completar con éxito varios despliegues comerciales en 2014.

SoftCOM es una visión revolucionaria de Huawei para la arquitectura de redes futuras basadas en conceptos clave y tecnología que incluyen la computación en la nube, SDN, NFV y operaciones basadas en Internet. A través de su estrategia SoftCOM, Huawei ayuda a las compañías de telecomunicaciones a implantar oportunidades estratégicas de la era de la información y gestionar de forma efectiva los retos existentes debidos a las limitaciones estructurales de las redes actuales.

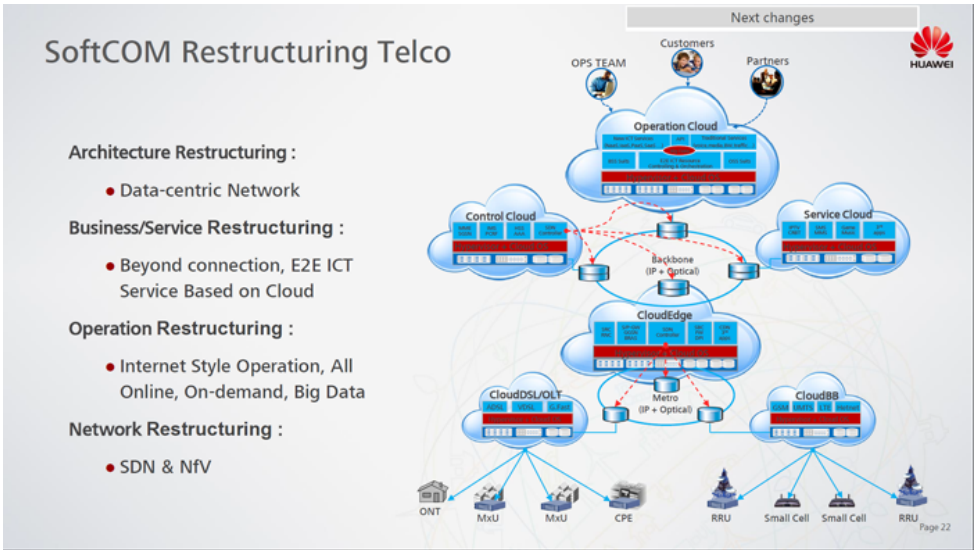


Figura 2.13 Esquema de la solución SoftCom propuesta por Huawei

2.5 Análisis de estándares, protocolos y proyectos.

El hecho de que las tecnologías SDN y NFV se estén extendiendo cada vez con más fuerza es en parte gracias a la aparición de estándares o protocolos que han facilitado su utilización.
La creación de estándares, especialmente los de código abierto (como por ejemplo los proyectos OpenFlow, OpenDayLight y OpenStack), ha supuesto para los desarrolladores y proveedores de servicios una "liberación". Esto se debe a que estos estándares o protocolos les permiten desarrollar nuevos servicios de red basados en aplicaciones, liberándose del software propietario de los dispositivos de red convencionales que limitan ese desarrollo [9].
A continuación se exponen algunos ejemplos de estándares o protocolos más importantes relacionados con estas tecnologías.

OpenFlow [20]
OpenFlow surgió a raíz del proyecto de Investigación: "OpenFlow: Enabling Innovation in Campus Networks" de 2008 en la Universidad de Stanford.
La primera versión del protocolo OpenFlow (version1.1) se lanzó en febrero de 2011 y la segunda (versión 1.2) fue supervisada por la ONF que tiene el control sobre la especificación. Actualmente la versión en vigor es la 1.4.0.
Algunos miembros fundadores son Google, Facebook y Microsoft, a los que se han ido añadiendo grandes empresas del sector de las telecomunicaciones como: Allied Telesis, Citrix, Cisco, Dell, HP, F5 Networks, IBM, NEC, Huawei, Juniper Networks, Oracle y VMware.
Además, la Open Networking Foundation (ONF) define OpenFlow como la primera interfaz de comunicaciones estándar entre las capas de control y de reenvío de una arquitectura SDN.

OpenFlow se define como un protocolo emergente de comunicaciones abierto que permite a un servidor de software determinar el camino de reenvío de paquetes que debería seguir en una red de switches. Concretamente, lo que ofrece OpenFlow es un protocolo abierto para poder programar las tablas de flujo en diferentes switches y routers.
La idea básica consiste en explotar el hecho de que la mayoría de los switches Ethernet contienen tablas de flujos (Flow-Tables). Aunque las tablas de flujos que utiliza cada fabricante son propias, se han aprovechado características que se ha visto

que son comunes a todas ellas.
Las acciones que pueden soportar los switches OpenFlow son extensibles, eso sí, es necesario tener unas características mínimas comunes a todos ellos.
No obstante los proveedores de red deben crear switches basados en OpenFlow con el fin de que OpenFlow se afiance.
OpenFlow traslada la decisión de determinar cómo se reenvían los paquetes de los switches a los controladores, normalmente un servidor o una estación de trabajo.

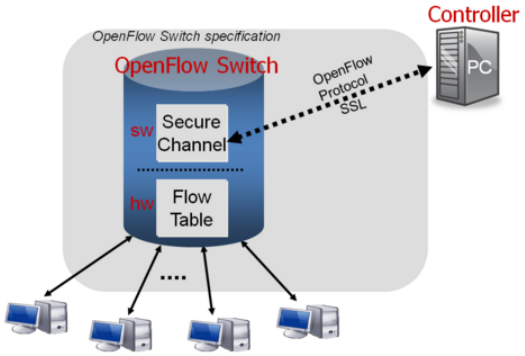


Figura 2.14 Especificación de switch OpenFlow

Una aplicación de gestión se ejecuta en las interfaces del controlador que une todos los switches en la red, facilitando la configuración de rutas de reenvío que utilizan todo el ancho de banda disponible. La especificación define el protocolo entre el controlador y los switches y un conjunto de operaciones que se pueden realizar entre ellos. Las instrucciones de reenvío se basan en el flujo, que consiste en que todos los paquetes comparten una serie de características comunes. Existen infinidad de parámetros que pueden especificarse para definir un flujo. Entre los posibles criterios se pueden incluir los puertos por donde se reciben los paquetes cuando llegan, el puerto Ethernet de origen, la etiqueta VLAN, el destino Ethernet o el puerto IP. Cuando un paquete que llega no encuentra ninguna coincidencia con ninguna entrada de la tabla se debe crear un nuevo flujo. El switch debería tener configurado un descartado de paquetes para el flujo que no haya sido definido, pero en la mayoría de los casos, el paquete se envía al controlador. El controlador entonces define un nuevo flujo para ese paquete y crea una o más entradas para la tabla. Posteriormente, este envía la entrada o entradas al switch para que sean añadidas a las tablas de flujo. Finalmente, el paquete se envía de vuelta al switch para ser procesado con las nuevas entradas que se han creado.
Dicho de otro modo, con el protocolo OpenFlow, una red puede ser gestionada como un todo, evitando tener que gestionar individualmente un gran número de dispositivos. Esto es posible gracias a que es el propio servidor el que dice a los switches donde deben enviar los paquetes. Con esta tecnología la red puede ser programada independientemente de los switches ya que las decisiones que implican el movimiento de paquetes están centralizadas. En los switches OpenFlow el reenvío de paquetes (datapath) y el encaminamiento de alto nivel (control path) se realizan de forma separada, mientras que en un switch convencional se realizan en el mismo dispositivo. Concretamente, con OpenFlow, una parte del datapath reside en el mismo switch, pero el que realiza las decisiones de encaminamiento de alto nivel es un controlador. Para comunicar ambos elementos se utiliza del protocolo OpenFlow.
Y es aquí donde se establece la relación entre este protocolo y SDN, ya que esta metodología, como ya se ha explicado en apartados anteriores, es conocida como SDN.
Viendo los beneficios que aporta OpenFlow basado en SDN, OpenFlow ha sido utilizado en ámbitos tan diversos como los centros de datos a gran escala, los centros de datos de empresas, proveedores públicos y privados de servicios en la red, redes de telecomunicaciones, redes de conmutación de circuitos, redes ópticas. También se

está utilizando para variedad de servicios de virtualización de redes, seguridad, control de acceso para balanceo de carga, Ingeniería de Tráfico y gestión de la energía.

OpenStack [21]

OpenStack es un proyecto gestionado por la Fundación OpenStack, creada en septiembre de 2012 para promover el software OpenStack y su comunidad.
El proyecto OpenStack es un proyecto de computación en la nube para proporcionar una infraestructura como servicio (IaaS) y se trata de un software libre y de código abierto distribuido bajo los términos de la licencia Apache.
Más de 200 empresas se unieron al proyecto, entre las cuales destacan AMD, Avaya, Brocade Communications Systems, Canonical, Cisco, Dell, Ericsson, Groupe Bull, HP, IBM, InkTank, Intel, NEC, Rackspace Hosting, Red Hat, SUSE Linux, VMware y Yahoo.
La tecnología consiste en una serie de proyectos relacionados entre sí que controlan pools de control de procesamiento, almacenamiento y recursos de red a través de un centro de datos (entre otros). Todos ellos administrados a través de un panel de control que permite a los administradores tener el control y potenciar a sus usuarios proveyéndolos de los recursos que necesitan a través de una interfaz web.
OpenStack tiene una arquitectura modular que está formada por los siguientes componentes:

- **Compute (Nova).** Es un controlador de estructura cloud computing, que es la parte principal de un sistema de IaaS.
- **Object Storage (Swift).** Es un sistema de almacenamiento redundante y escalable.
- **Block Storage (Cinder).** Proporciona dispositivos de almacenamiento a nivel de bloque persistentes para usar con instancias de OpenStack Compute.
- **Networking (Neutron, anteriormente Quantum).** Es un sistema para la gestión de redes y direcciones IP.
- **Dashboard (Horizon).** Proporciona a los administradores y usuarios una interfaz gráfica para el acceso, la provisión y automatización de los recursos basados en la nube.
- **Servicio de Identidad (Keystone).** Ofrece un directorio central de usuarios asignados a los servicios de OpenStack que pueden acceder.
- **Servicio de Imagen (Glance).** Proporciona servicios de descubrimiento, de inscripción y de entrega de los discos y del servidor de imágenes.
- **Telemetría (Ceilometer).** Proporciona un único punto de contacto para los sistemas de facturación, proporcionando todos los contadores que se necesitan para establecer la facturación del cliente, a través de todos los componentes actuales y futuras de OpenStack.
- **Orquestación (Heat).** Es un servicio para orquestar múltiples aplicaciones compuestas en la nube utilizando plantillas, tanto a través de una API REST OpenStack nativa y una API de consultas compatibles con CloudFormation.
- **Base de datos (Trove).** Es una base de datos que funciona como un servicio de aprovisionamiento de motores de bases de datos relacionales y no relacionales.

Pero este estudio se centrará en el componente OpenStack Networking (Neutron, anteriormente Quantum). OpenStack Networking (Neutron) es un sistema para la gestión de redes y direcciones IP que permite asegurar que la red no presente el problema del cuello de botella o el factor limitante en un despliegue en la nube. Además de esto ofrece a los usuarios un autoservicio real, incluso a través de sus configuraciones de red.

OpenStack Networking proporciona unos modelos de redes para diferentes aplicaciones o grupos de usuarios. Los modelos estándar incluyen redes planas o VLAN para la separación de los servidores y el tráfico. Además de gestionar las direcciones IP, lo que permite direcciones IP estáticas o DHCP reservadas. Por otro lado, las direcciones IP flotantes permiten que el tráfico se redirija dinámicamente a cualquiera de sus recursos informáticos, lo que permite redirigir el tráfico durante la realización de un mantenimiento o en caso de fracaso.
Los usuarios pueden crear sus propias redes, controlar el tráfico y conectar los servidores y los dispositivos a una o más redes. Los administradores pueden conseguir altos niveles de multiempresa y escala masiva gracias a que OpenStack Networking permite aprovechar las redes definidas por software de tecnología (SDN), por ejemplo con OpenFlow.
Además, OpenStack Networking tiene un framework que permite la extensión de servicios de red adicionales, como los sistemas de detección de intrusos (IDS), balanceo de carga, firewalls y redes privadas virtuales (VPN).

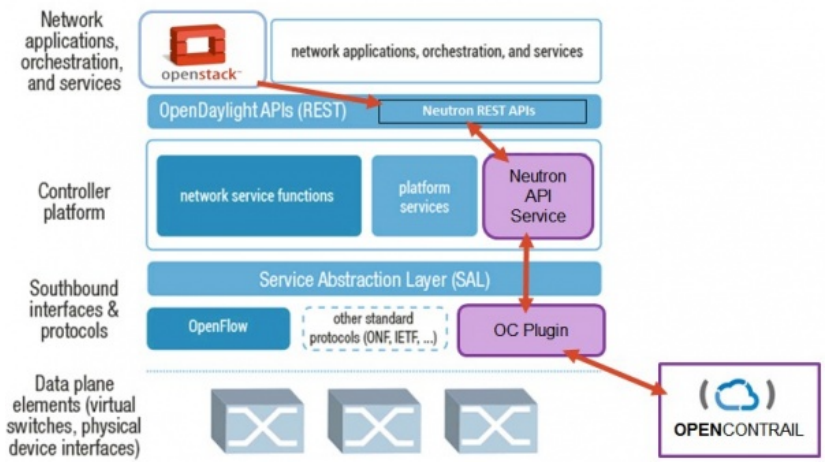


Figura 2.15 Esquema de OpenStack Neutron

OpenDaylight [22]
El Proyecto OpenDaylight es un proyecto de colaborativo de código abierto organizado por la Fundación Linux cuyo objetivo es acelerar la adopción de redes definidas por software (SDN) y crear una base sólida para las funciones de red de virtualización (NFV). El software está escrito en Java.
En abril de 2013, La Fundación Linux anunció la fundación del Proyecto OpenDaylight dirigido por la comunidad y apoyado por la industria de framework de código abierto para acelerar la adopción, fomentar la innovación y crear un enfoque más abierto y transparente a Software-Defined Networking (SDN) y las funciones de red de virtualización (NFV). Los miembros fundadores del proyecto (Arista Networks, Big Switch Networks, Brocade, Cisco, Citrix, Ericsson, HP, IBM, Juniper Networks, Microsoft, NEC, Nuage Networks, PLUMgrid, Red Hat and VMware) se comprometieron a donar recursos software y de ingeniería para el framework de código abierto OpenDaylight para ayudar a definir el futuro de una plataforma SDN de código abierto.

Mediante el apoyo a los estándares abiertos, como OpenFlow Networking Standard, OpenDaylight entregó un framework común de código abierto y una plataforma para SDN a través de toda la industria para los clientes, socios y desarrolladores. El primer código del proyecto OpenDaylight, llamado Hydrogen, fue lanzado en febrero de 2014. Las esperadas donaciones y proyectos para Hydrogen incluyen un controlador abierto, una red virtual superpuesta y mejoras en los switches.

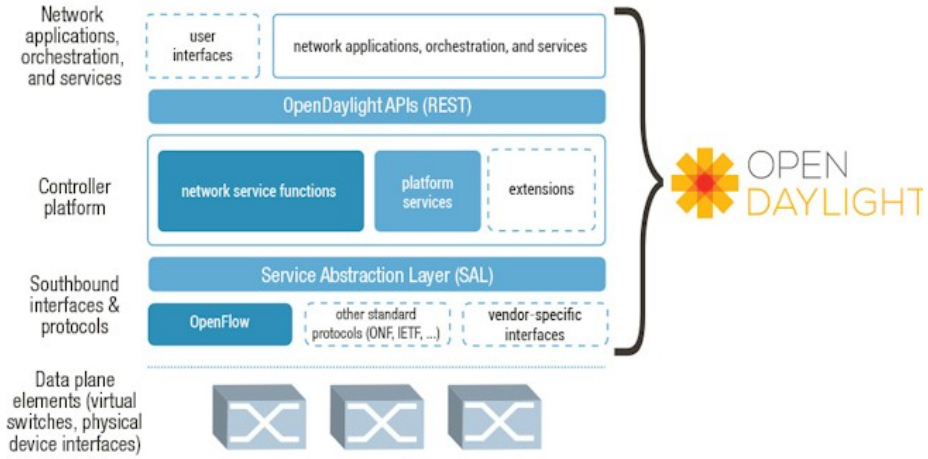


Figura 2.16 Esquema de OpenDaylight

La Open Networking Foundation (ONF) define OpenFlow como la primera interfaz de comunicaciones estándar entre las capas de control y de reenvío de una arquitectura SDN.
A pesar de esto y de todas las ventajas que aporta OpenFlow, también se plantean una serie de desafíos, que van desde la escalabilidad a la seguridad. Pero lo más destacable es que los proveedores de red deben desarrollar switches basados en OpenFlow para que este se afiance en la industria.
Aunque la mayoría de los proveedores de red ya han desarrollado equipos basados en OpenFlow, por otro lado también están diseñando arquitecturas SDN que utilizan métodos de comunicación alternativos (incluyendo protocolos de red existentes, tales como MPLS y NETCONF.
A continuación se explican algunos protocolos SDN alternativos a los tres anteriores, quizá no tan extendidos ni conocidos pero que también se pueden utilizar.

Border Gateway Protocol (BGP) [23]
Border Gateway Protocol (BGP) es un protocolo que se utiliza en las redes de sistemas autónomos para intercambiar información de enrutamiento entre gateways. Este protocolo se considera un protocolo de gateway exterior estandarizado y se utiliza frecuentemente entre gateways en internet. BGP a menudo también se clasifica como un protocolo de vector de distancia o un protocolo de ruta vectorial. En la red, cada gateway tiene normalmente su propio router. La tabla de enrutamiento contiene una lista de routers conocidos, las direcciones que pueden alcanzar, y una métrica de coste asociado con la ruta a cada router. Todo esto permite elegir la mejor ruta disponible.
Los proveedores están intentando utilizar BGP en redes definidas por software híbridas. Algunos argumentan que el protocolo sothbound en una arquitectura SDN es menos importante que la agilidad operativa y la capacidad de programación que ofrece SDN, con o sin OpenFlow. Como resultado, los proveedores identificaron a BGP como un protocolo SDN con potencial para conseguir la programabilidad de red prometida por SDN.

NETCONF [23]
NETCONF es un protocolo de gestión de red Internet Engineering Task Force (IETF). Permite a un administrador de red configurar de forma segura cualquier dispositivo de red, como por ejemplo un firewall, un switch o un router. Se basa en la llamada de procedimiento remoto (RPC) y fue diseñado para resolver los problemas que existen con los Protocolos de Gestión de Red Simple y la Interfaz de Comandos en Línea.
La Fundación Open Networking (ONF) aceptó recientemente NETCONF y lo hizo

obligatorio para la configuración de dispositivos compatibles con OpenFlow. La especificación, llamada OF-CONFIG, requiere que los dispositivos que la soportan deban implementar el protocolo NETCONF como el transporte.

Protocolo de Presencia y Mensajería Extensible (XMPP) [23]

El Protocolo de Presencia y Mensajería Extensible (XMPP) es un protocolo basado en XML (del inglés Extensible Markup Language). Originalmente su uso estaba pensado para mensajería instantánea y para la detección de presencia en línea.
El protocolo funciona entre los servidores y facilita la operación en tiempo casi real. XMPP ha surgido recientemente como un protocolo SDN alternativo a OpenFlow en redes SDN híbridas. Puede ser utilizado por el controlador para distribuir información tanto del plano de control como del plano de gestión a los puntos finales de servidor.
Gestiona la información en todos los niveles de abstracción, hasta llegar al flujo.

Protocolo de Gestión de Base de Datos Open vSwitch (OVSDB) [23]

El Protocolo de Gestión de Base de Datos Open vSwitch (OVSDB) es un protocolo de configuración de OpenFlow que está destinado a administrar las implementaciones Open vSwitch. Open vSwitch es un switch virtual que permite la automatización de la red y el soporte de interfaces y protocolos de administración estándar, como por ejemplo NetFlow. El protocolo también soporta la distribución a través de múltiples servidores físicos.
En una implementación Open vSwitch, un grupo de control contiene administradores y controladores que utilizan el protocolo OVSDB para suministrar información de configuración al servidor de base de datos de switches. Los controladores usan OpenFlow para especificar los detalles de los flujos de paquetes a través del switch. Cada administrador y controlador pueden dirigir varios switches, y cada switch puede recibir directivas de varios gestores y controladores.

Perfil de Transporte MPLS (MPLS-TP) [23]

Perfil de Transporte MPLS (MPLS-TP) es el perfil de transporte para la conmutación de etiquetas multiprotocolo. MPLS-TP está diseñado para ser utilizado en redes de transporte como una tecnología de capa de red. Las extensiones del protocolo MPLS están siendo diseñadas por el IETF y para ello se está basando en los requerimientos proporcionados por los proveedores de servicios. El protocolo es una aplicación de conmutación de paquetes, orientado a la conexión (CO-PS), que ofrece una implementación MPLS dedicada a eliminar características que no son relevantes a las aplicaciones CO-PS y a agregar dispositivos que proporcionan soporte a la funcionalidad de transporte crítico.
La Fundación Open Networking propuso cambios a MPLS que incluían el uso de los datos de plano estándar MPLS con un plano de control más simple basado en SDN y OpenFlow. Al contar con un plano de control simplificado que se desacopla del plano de datos, la ONF argumenta que es capaz de optimizar los servicios globalmente, hacerlos más dinámicos y crear nuevos programando aplicaciones de red sobre el controlador SDN.

3. Capítulo 3, Comparativa de las tecnologías SDN y NFV.

Llegados a este punto ya se conocen cuáles han sido los orígenes de estas tecnologías, porque han surgido y en que consisten. Así que se dispone de la información necesaria para poder completar el estudio de estas tecnologías realizando una comparativa de ambas [8], [10], [25], [26], [27].

3.1 Puntos en común o similitudes.

Hay que partir de la base de que, como ya se ha visto en los capítulos anteriores, SDN y NFC son tecnologías diferentes. No obstante, una vez dicho esto, SDN y NFV son conceptos muy relacionados entre sí y ambos pueden ser complementarios, aunque pueden perfectamente existir el uno sin el otro. Incluso en algunos entornos se supone erróneamente que SDN y NFV están "enfrentados", cuando en realidad tienen más en común e incluso trabajan juntos y se complementan.
Dicho de otro modo, NFV es altamente complementario con SDN, pero no depende de ella (o viceversa). NFV se puede implementar sin que se requiera SDN, aunque los dos conceptos y soluciones se pueden combinar consiguiendo así un mayor potencial y adaptación a las nuevas necesidades tecnológicas, tal y como se muestra en la figura 3.1 [25]

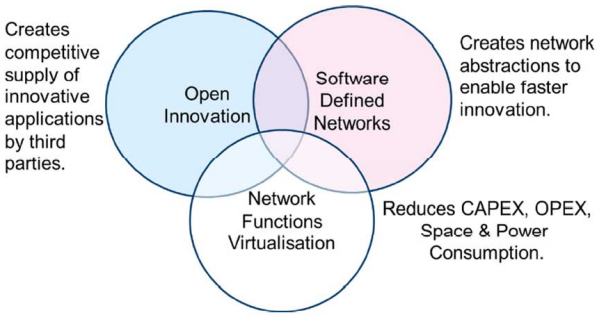


Figura 3.1 Relación de NFV con SDN

NFV puede conseguir sus objetivos utilizando mecanismos no SDN, basándose en las técnicas que se están utilizando actualmente en muchos centros de datos. Pero los enfoques basados en la separación de los planos de control y de reenvío de datos (como propone SDN) pueden mejorar el rendimiento, simplificar la compatibilidad con las implementaciones existentes y facilitar los procedimientos de operación y mantenimiento. NFV es capaz de soportar SDN, proporcionando la infraestructura sobre la que el software SDN se puede ejecutar. Además, las funciones de red virtualizadas se alinean estrechamente con los objetivos SDN de utilizar servidores y switches básicos.

Otra cosa que tienen en común las tecnologías SDN y NFV es que, a pesar de que ambas son tecnologías "jóvenes", son el inicio de una nueva tendencia en networking, cuyo objetivo es, principalmente, adecuar la actual arquitectura de redes a las nuevas exigencias tecnológicas y a las nuevas necesidades surgidas de la constante evolución tecnológica, simplificando la gestión de las redes.

Tanto la Virtualización de funciones de red (NFV) como las redes definidas por software (SDN) pretenden alejarse del hardware y acercarse al software. Debido a esto, estas tecnologías están cambiando significativamente la relación entre los proveedores de servicios y los proveedores de equipos. Los operadores empiezan a no aceptar sistemas cerrados y propietarios controlados por proveedores de equipos,

y en cambio están exigiendo el control de sus redes para aumentar la innovación, flexibilidad y capacidad de respuesta necesarias para enfrentar los requisitos actuales.

Por otro lado, tanto SDN como NFV pretenden avanzar en un enfoque basado en software para la creación de redes con el objetivo de hacer unas redes más escalables, ágiles e innovadoras que puedan alinearse y apoyar los objetivos de TI globales del negocio.
Por lo tanto, no es de extrañar que algunos conceptos comunes guíen el desarrollo de cada uno, por ejemplo, cada uno de ellos tienen como objetivos principales:

- Desplazar la funcionalidad al software.
- Utilizar servidores y switches básicos en lugar de dispositivos propietarios reduciendo así la dependencia hardware.
- Aprovechar las Interfaces de Programación de Aplicaciones (API).
- Apoyar la orquestación, virtualización y automatización más eficiente de los servicios de red.

En cuanto a sus objetivos, tanto SDN como NFV tienen como prioridad controlar la red lógicamente (con software) y minimizar el trabajo con los dispositivos de red (hardware). Es decir, ambas dan un enfoque basado en software para la creación de redes, consiguiendo así una mayor agilidad y flexibilidad además de permitir reducir CAPEX y OPEX.
Dicho de otro modo, ambas tecnologías están orientadas a intentar satisfacer algunas de las necesidades de los operadores, como son:

- Poder reducir las inversiones destinadas a CAPEX y OPEX.
- Poder entregar al usuario final servicios de una forma más rápida y dinámica.
- Poder evolucionar hacia redes más escalables e innovadoras.

3.2 Diferencias

Como ya se ha dicho anteriormente, SDN y NFV están muy relacionadas la una con la otra y en muchos casos se utilizan conjuntamente ya que la combinación de ambas permite obtener mayores beneficios. No obstante, una vez dicho esto, no hay que olvidar que en el fondo se está hablando de dos tecnologías diferentes y por lo tanto existen una serie de diferencias entre ellas.

La primera diferencia se encuentra en su origen, mientras que SDN fue creada por investigadores y arquitectos de centros de datos, NFV fue creada por un consorcio de proveedores de servicios.
Siguiendo con su origen, SDN se creó antes que NFV, lo que hace que esté algo más desarrollada y extendida.

A la hora de implementar estas tecnologías, en el nivel más básico, ambas necesitan una red. Pero la diferencia reside en que NFV es más versátil ya que la red que requiere para ejecutarse puede ser una red tradicional basada en hardware (incluso si la red ya existe, no ha de ser nueva), una red SDN, o una combinación de ambas. En cambio, SDN requiere de la construcción de una nueva red donde los planos de datos y de control estén separados.

Otra diferencia importante entre ambas es que tratan ámbitos diferentes. Por un lado SDN se centra en la separación de la capa de control de la capa de datos, para permitir la creación de redes virtuales. Mientras que NFV se centra en la virtualización de las funciones de la red, para permitir la migración de un esquema de red basado en nodos hardware específicos, a otro basado en hardware de propósito general e incluso

cloud.
En el apartado de similitudes ya se ha comentado que tanto SDN como NFV permiten reducir CAPEX y OPEX y proporcionan agilidad y flexibilidad. Pero existe una diferencia entre ambas. Mientras que NFV permite acelerar el tiempo de salida al mercado, SDN facilita la innovación.

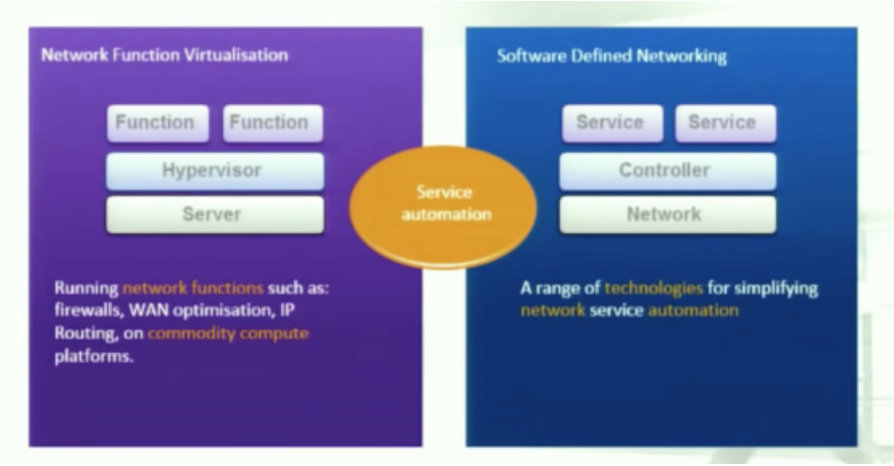


Figura 3.2 Arquitecturas y finalidades de NFV y SDN

En cuanto a su funcionamiento y objetivos se puede observar que también existen algunas diferencias.
Por un lado, SDN introduce la centralización del plano de control y programabilidad de la red instalando el software en un controlador. De esta manera consigue:

- Que las funciones de red se puedan configurar más fácilmente.
- Que se pueda llevar a cabo una distribución de servicios en la nube más eficiente.

Es decir, ofrece un funcionamiento lógico de la red de acuerdo a la demanda dinámica.
Además, como los dispositivos de la capa de datos ya no tienen que tomar ninguna decisión con los paquetes que reciben, ya no es necesario programar cada terminal porque esta función la efectúa el controlador. Esto hace que se pueda recudir el gasto en dispositivos de la capa de control.
En cambio, NFV tiene en cuenta la instalación de las funciones de red en dispositivos software virtualizados. Así, sólo requiere la implementación de hipervisores que proporcionen la suficiente abstracción del hardware para conseguir la portabilidad en las aplicaciones de software.

También existe una diferencia importante entre SDN y NFV a la hora de hablar de los protocolos que utilizan. Por un lado SDN utiliza el protocolo OpenFlow. En cambio, en el caso de NFV, la ETSI (organización encargada de su estandarización) de momento no ha creado ninguno.

3.3 Resumen comparativo

La siguiente tabla intenta ofrecer de forma resumida una breve comparación de algunos de los puntos más importantes de las tecnologías SDN y NFV [25].

Categoría	SDN	NFV
Origen	Creado por investigadores y arquitectos de centros de datos	Creado por un consorcio de proveedores de

		servicios
Objetivo principal	Enfoque basado en software para la creación de redes	Enfoque basado en software para la creación de redes
Razón de ser	Separación de control y datos, centralización del control y capacidad de programación de la red	Reubicación de las funciones de red de dispositivos dedicados a servidores genéricos
Localización de destino	Campus, data center / cloud	Red del proveedor de servicios
Tipo de red	Red donde los planos de datos y de control estén separados	Red tradicional basada en hardware, una red SDN, o una combinación de ambas
Dispositivos de destino	Servidores y switches básicos	Servidores y switches básicos
Aplicaciones iniciales	Orquestación del cloud y las redes	Routers, firewalls, gateways, CDN, aceleradores WAN,…
Nuevos protocolos	OpenFlow	Ninguno todavía
Formalización	Open Networking Forum (ONF)	ETSI NFV Working Group
Iniciador de negocio	Organización IT	Proveedor de servicios
Beneficios para el cliente	Reduce CAPEX y OPEX, proporciona agilidad y flexibilidad y facilita la innovación	Reduce CAPEX y OPEX, proporciona agilidad y flexibilidad y acelera el tiempo de salida al mercado

Tabla 3.1 Comparación de los puntos clave de SDN y NFV

4. Capítulo 4, Creación de un escenario virtual.

En los apartados anteriores se han estudiado, de forma teórica, las tecnologías SDN y NFV. Ahora, para acabar de completar su estudio, se procederá a la implementación de dos casos prácticos a modo de ejemplo.
Concretamente en este capítulo se recreará un escenario SDN virtual.

4.1 Elección del escenario a recrear.

Se decide crear un escenario virtual que emule una red SDN con openFlow. Para ello se utiliza Mininet. Se ha escogido Mininet para implementar el escenario porque es una plataforma que permite crear redes virtuales a gran escala de forma rápida y eficiente. Es decir, es un emulador de red que ejecuta una colección de dispositivos finales, switches, routers y enlaces en un solo core de Linux. Esto lo consigue gracias al uso de una característica conocida como virtualización ligera (light-weight virtualization). Debido a que Mininet permite implementar nodos con el protocolo OpenFlow, es muy práctico para el estudio de SDN.
Las principales características de Mininet son:

- Es un simulador de SDN.
- Corre sobre Linux.
- Sólo se necesita un PC y los prototipos se pueden escalar a redes grandes.
- Bueno para crear prototipos. El comportamiento del prototipo representa el comportamiento real con un alto grado de realismo.
- Permite crear topologías personalizadas mediante scripts (en Python).
- Para la simulación emula los diferentes enlaces, PCs, switches y controladores.
- Puede tener hasta cientos de nodos, dependiendo de la configuración.
- La administración y la simulación ocurren en tiempo real.
- Permite la reutilización de componentes (scripts).

4.2 Implementación del escenario.

Se ha utilizado Mininet VM 2.2.1 (image) para implementar el escenario SDN. Esta versión se basa en el sistema operativo Ubuntu Server 14.04 64 bits y viene con todo el software y las herramientas necesarias para apoyar al Mininet que ya está instalado. Esta versión se puede importar fácilmente en cualquier virtualizador (VMWare, VirtualBox,…) y crear una máquina virtual con Mininet, openFlow, wireshark,… Esto hace que sea una opción muy ágil para montar el escenario SDN.
Pero antes de poder crear el escenario SDN definitivo con Mininet-VM hay que hacer una serie de tareas para poder poner el entorno en marcha.
Seguidamente se explican los pasos que se han seguido para la creación del escenario SDN con Mininet.

Creación y configuración de la máquina virtual con VirtualBox [34], [38]
El primer paso es la creación de la máquina virtual Mininet-VM descargando de la web de mininet la versión indicada anteriormente [31].

Para crear la máquina virtual se ha utilizado VirtualBox 5.0.4 sobre un OS X 10.11 (El Capitán). Así que una vez finalizada la descarga se ejecuta el archivo .ovf y empieza automáticamente la importación en VirtualBox [32].
A continuación se aceptan las opciones que muestra VirtualBox y en breve ya se dispone de la máquina virtual Mininet-VM. Pero aún falta hacer algunas configuraciones para que esta funcione correctamente.

Lo primero es añadir un adaptador de “sólo-anfitrión” en VirtualBox, según se recomienda en las notas de configuración de Mininet [33]. Esto crea una interfaz de bucle invertido en el equipo host que se puede utilizar para conectar la máquina virtual al ordenador principal.
Para hacerlo se abre el panel de preferencias de VirtualBox (VirtualBox → Preferencias), se hace clic en el icono de "Red" del panel de Preferencias y posteriormente en la etiqueta “redes sólo-anfitrión”. Entonces, se hace clic en el pequeño icono verde "más" de la parte derecha de la ventana para agregar un nuevo adaptador de red. Esto crea un adaptador llamado “vboxnet0”, cuya configuración por defecto ya es adecuada.

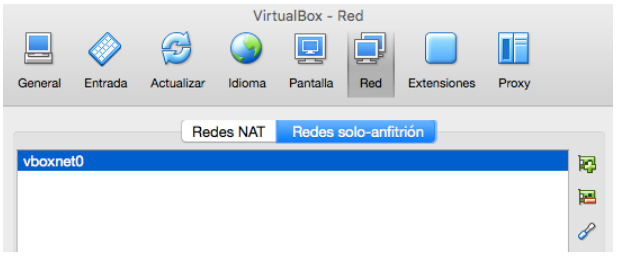


Figura 4.1 Añadir adaptador “vboxnet0”

A continuación se comprueban los ajustes haciendo clic en el pequeño icono de un "destornillador" de la parte derecha de la ventana. Se observa que el servidor DHCP está habilitado en la interfaz y se ve que la dirección inferior es 192.168.56.101/24. Así, se sabe que la dirección IP de la interfaz virtual conectada a la red de “sólo-anfitrión” en la máquina virtual que se asigna es esa dirección IP.
El siguiente paso es agregar un adaptador de red a la máquina virtual Mininet.
En la ventana del administrador de VirtualBox, se hace clic en la máquina virtual Mininet-VM y luego en el icono "Configuración" en la parte superior de la ventana. Después se clica en el icono de "Red" en el panel de configuración que aparece. Se observa que la máquina virtual ya tiene definida una interfaz en la pestaña "Adaptador 1", configurada como NAT (para conectarse a internet). Pero aún se necesita un camino para que la máquina virtual pueda conectarse directamente al ordenador anfitrión. Así que se añade otro adaptador virtual y se conecta a la interfaz “sólo-anfitrión” que se ha creado anteriormente.
Para ello se hace clic en la pestaña "Adaptador 2" y en el desplegable "Conectado a:" se selecciona “Adaptador sólo-anfitrión”. Esto permite que otros programas que se ejecutan en el ordenador anfitrión puedan conectarse a la máquina virtual utilizando SSH.

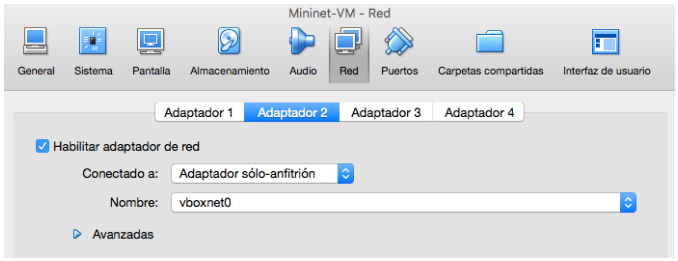


Figura 4.2 Añadir adaptador “sólo-anfitrión”

Llegados a este punto ya se ha realizado toda la configuración necesaria en VirtualBox y se puede proceder a iniciar la máquina virtual Mininet-VM.

Arrancar la máquina virtual Mininet-VM [34], [38]
Lo primero es iniciar la máquina virtual Mininet-VM, para ello se abre el gestor de

VirtualBox, se selecciona la máquina virtual Mininet-VM y luego se hace clic en el botón "Inicio". La máquina virtual se iniciará y se presentará una pantalla de login (usuario = mininet y contraseña = mininet).
Ya se tiene la máquina virtual en marcha y se está loguinado, pero aún es necesario configurar una nueva interfaz ethernet virtual. Para ello se ejecuta el comando ifconfig, que permite ver las interfaces disponibles en la máquina virtual Mininet-VM. Se observan dos interfaces: eth0 y lo. Previamente se ha configurado la máquina virtual Mininet-VM de modo que la interfaz eth0 se conecte a la interfaz NAT en el equipo anfitrión, por lo que se ha asignado una dirección IP que va a trabajar en la LAN conectada a este equipo. En este caso, eth0 tiene la dirección IP 10.0.2.15/24.
Esto permite que la máquina virtual se conecte a internet.
Pero en cambio, no se observa ninguna interfaz que se conecte a la interfaz de “sólo-anfitrión”. Es necesario configurar eth1.
Para ver todas las interfaces disponibles, incluso las que no están activas, se ejecuta el comando ifconfig -a.

```
mininet@mininet-vm:~$ ifconfig -a
```

Ahora se ve que hay tres interfaces: lo, eth0 y eth1. Pero eth1 no está activa y no tiene una dirección IP asignada. Así que se inicia el cliente DHCP en eth1, esto hará que solicite una dirección IP al servidor DHCP que se ejecuta en la interfaz “sólo-anfitrión” conectada a eth1.

```
mininet@mininet-vm:~$ sudo dhclient eth1
```

Si se ejecuta el comando ifconfig de nuevo, se observa que la interfaz eth1 ahora tiene asignada la dirección IP 192.168.56.101.
Se puede ver que esta interfaz está operativa haciendo ping a la dirección de interfaz de “sólo-anfitrión” del equipo host (192.168.56.1) de la máquina virtual Mininet-VM.
Para configurar de forma permanente la nueva interfaz y que al iniciar la máquina virtual el sistema configurare automáticamente la interface eth1, se edita el archivo /etc/network/interfaces y se añade una interfaz llamada eth1. Sólo hay que copiar la configuración de eth0 para eth1. En este caso, se han añadido las siguientes líneas al final del archivo:

```
auto eth1
iface eth1 inet dhcp
```

En este punto la máquina virtual Mininet-VM ya está configurada.
El siguiente paso es conectarse a esta máquina virtual desde el equipo anfitrión mediante SSH.

Uso de SSH para conectarse a la MiniNet-VM [34], [38]

A continuación se va a utilizar el software de cliente SSH en el ordenador anfitrión para conectarse a la máquina virtual Mininet. Con esto se logran dos cosas:

- Desde el equipo anfitrión, se pueden establecer conexiones a las aplicaciones X remotas que se ejecutan en la máquina virtual Mininet-VM, como xterm y Wireshark.
- Se puede utilizar una ventana de terminal más amigable o un terminal xterm para interactuar con la máquina virtual Mininet-VM. Esto facilita el trabajo ya que trabajar con la ventana de la consola VirtualBox es difícil porque:
    - La ventana de la consola VirtualBox captura el puntero del ratón cada vez que se utiliza.
    - No se puede cortar y pegar texto desde la ventana de la consola de la máquina virtual a un programa en el equipo anfitrión.

Para establecer la conexión SSH se abre la aplicación Terminal en el ordenador

anfitrión y mediante el cliente SSH se inicia una sesión de SSH a la máquina virtual Mininet-VM.

```
MacBook-Air:~ Fran$ ssh -Y mininet@192.168.56.101
```

A continuación aparece una advertencia acerca de la clave RSA, se introduce "yes" para continuar. Esto añade la clave RSA al cliente SSH para que no aparezca el aviso de nuevo.
Ahora se está conectado a la máquina virtual Mininet-VM a través de una sesión SSH segura y se pueden ejecutar clientes X11 remotos en la máquina virtual Mininet-VM sobre esta sesión SSH. Pero en este caso, al ser el equipo anfitrión un MAC es necesario tener instalado el servidor XQuartz X, en este caso está instalada la versión 2.7.8 [35].
En la ventana de terminal, en la que ahora se está ejecutando la sesión SSH conectada a la máquina virtual Mininet-VM, se inicia un xterm con el comando xterm -sb &.

```
mininet@mininet-vm:~$ xterm -sb &
```

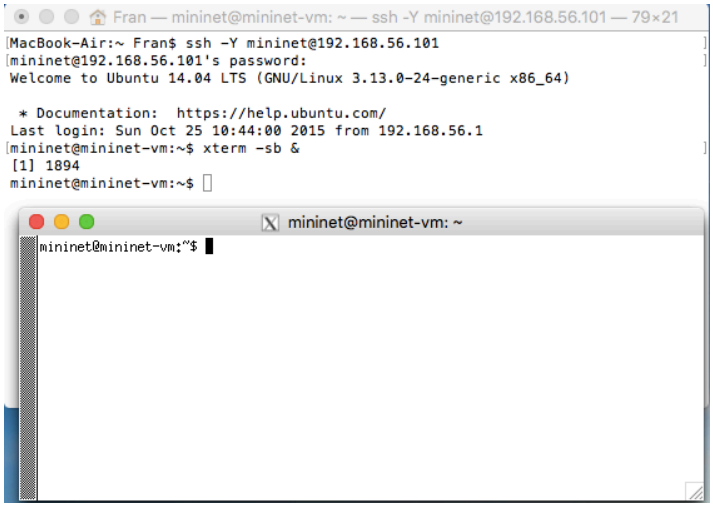


Figura 4.3 Conexión SSH a Mininet-VM y apertura de un Xterm

A continuación, tal y como se muestra en la figura 4.3 se abrirá una ventana Xterm. La opción "-sb" permite tener un buffer de desplazamiento en la ventana Xterm, para poder desplazarse hacia atrás si es necesario.
Una vez realizadas todas estas tareas, ya se dispone de la máquina virtual Mininet-VM completamente configurada y una conexión SSH con un Xterm en el que poder lanzar los comandos necesarios.

Creación del escenario SDN [34], [38]

A continuación se describen las opciones más comunes para iniciar mininet.
Para crear una topología mínima y entrar en el CLI de mininet se utiliza el comando mn.

```
mininet@mininet-vm:~$ sudo mn
```

La topología predeterminada de mininet es la topología “mínima”, que incluye un switch con kernel de OpenFlow conectado a dos PCs (hosts) más el controlador de referencia OpenFlow.
Algunos comandos básicos para empezar a utilizar mininet son:

Comando	Descripción
sudo mn -c	Elimina los posibles “restos” de una topología de mininet anterior
sudo mn -h	Ayuda de mininet

nodes	Muestra los nodos de la topología
net	Muestra los enlaces de la topología
dump	Muestra la información de todos los nodos de la topología
pingall	Hace ping entre todas las parejas de nodos de la topología

Tabla 4.1 Comandos más habituales de Mininet

Llegados a este punto se crea una topología básica para comprobar que todo funciona correctamente.

Figura 4.4 Creación de una topología básica

En la figura 4.4 se puede observar como se crea la topología básica:

- un switch (s1)
- dos hosts (h1 y h2)
- un controlador (c0)
- dos enlaces (h1 con s1 y h2 con s1)

Con los comandos del CLI de mininet explicados anteriormente se comprueba que se han creado todos los elementos, su conectividad,…

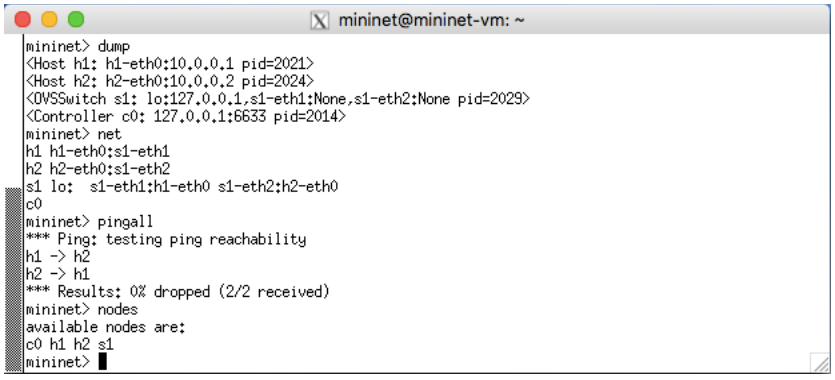


Figura 4.5 Comprobación de creación de los elementos y su conectividad

Por otro lado, como ya se ha comentado anteriormente, esta máquina virtual viene con algunas herramientas de soporte, como por ejemplo Wireshark.
Wireshark es una herramienta que permite ver el tráfico de control utilizando el "OpenFlow Wireshark disector".
Para ello hay que abrir Wireshark en segundo plano con el comando sudo wireshark &. Esto es conveniente hacerlo desde la ventana Xterm que se ha abierto anteriormente.

```
mininet@mininet-vm:~$ sudo wireshark &
```

A continuación se abre la aplicación Wireshark y se introducen una serie de parámetros.
Lo primero es indicar el tipo de filtro, para ello se introduce “of” en el apartado del filtro y se hace clic en Apply. Esto captura los paquetes OpenFlow.
Después, se hace clic en Capturar → Interfaces y a continuación se selecciona la interfaz loopback (lo) y se clica en Start.

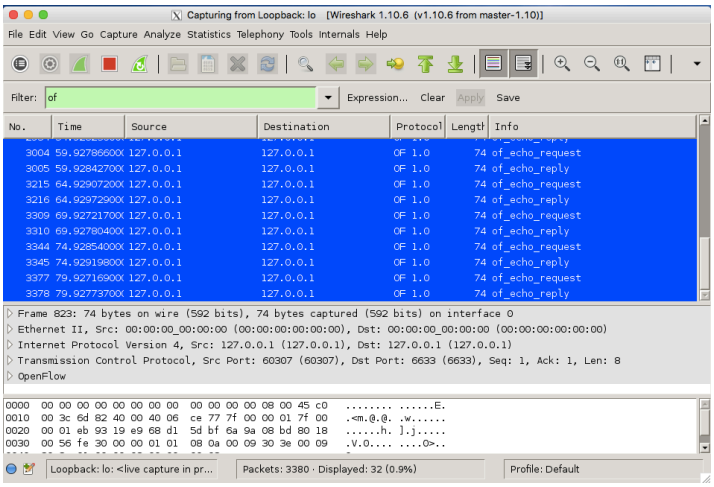


Figura 4.6 Wireshark capturando paquetes OpenFlow

Si se inicia primero Wireshark, se observa que no aparen paquetes OpenFlow en la ventana principal. Pero, si a continuación se crea la topología con mininet, se puede observar como empiezan a aparecer paquetes (Figura 4.6).

Llegados a este punto se puede decir que se dispone de un entorno SDN virtualizado operativo, sobre el cual se pueden implementar los casos de uso.

Comprobación del entorno SDN

Pero antes de pasar a implementar los casos de uso, para comprobar que el entorno es completamente operativo, se procede a ejecutar algunos ejemplos. Primero unos que utilizan la API de Mininet y después otros que se basan en la implementación de componentes de controladores. Estas pruebas también servirán para poder decidir cual de las dos opciones se utilizará en la implementación de los casos de uso.

Scripts que utilizan la API de Mininet

Se ejecutan unos scripts de ejemplo, escritos en python, que vienen con mininet (/mininet/examples) [38]. Para ejecutarlos basta con teclear el siguiente comando desde la ventana Xterm:

mininet@mininet-vm:~$ sudo mininet/examples/archivo.py [36]

Se han ejecutado los siguientes ejemplos y todos han funcionado correctamente:

- treeping64.py
- multiping.py
- mobility.py
- linearbandwidth.py
- miniedit.py
- consoles.py

Estos scripts utilizan la API de Mininet, se programan en python, son relativamente fáciles de programar y permiten hacer ciertas cosas útiles. Pero en general, las

funcionalidades que se pueden implementar no están totalmente relacionadas con la filosofía de SDN, programar o automatizar la red. Como se ha visto en los ejemplos, no se utilizan controladores para programar los switches, no se crean flujos de forma automática,…

Componentes de controladores remotos [41]

No obstante, también existe la opción de lanzar un controlador remoto que ejecute un componente propio. Pero si no se indica explícitamente, Mininet lanza el controlador que incorpora por defecto, que actúa como un switch clásico. En terminología OpenFlow, se le denomina Ethernet Learning Switch. Su funcionamiento es el de un switch tradicional: irá creando una tabla donde se asociarán direcciones MAC con puertos según vaya recibiendo tramas. Y a continuación enviará un mensaje OpenFlow al switch, creando una nueva entrada en la tabla de flujo.
Este controlador no admite componentes creados por el usuario, pero es útil para entender el flujo de mensajes y la separación del plano de decisión y el plano de conmutación.
El parámetro para poder seleccionar otro controlador es --controller [40]. Por ejemplo:

```
mininet@mininet-vm:~$  sudo mn --topo single, 3 --controller remote
```

La sintaxis general del parámetro --controller es:
--controller remote, ip=[controller IP], port=[controller port]

Esto permitiría instalar un controlador en cualquier lugar, pero si no se especifican, se toman los valores IP=127.0.0.1 y port=6633. Estos valores se corresponden con la máquina virtual, por lo que en este caso no es necesario especificarlos.
Para comprobarlo se ejecuta el comando:

```
mininet@mininet-vm:~$sudo mn --topo single,3 --switch ovsk --controller remote
```

El parámetro --switch ovsk indica que se utilice un switch OpenFlow.

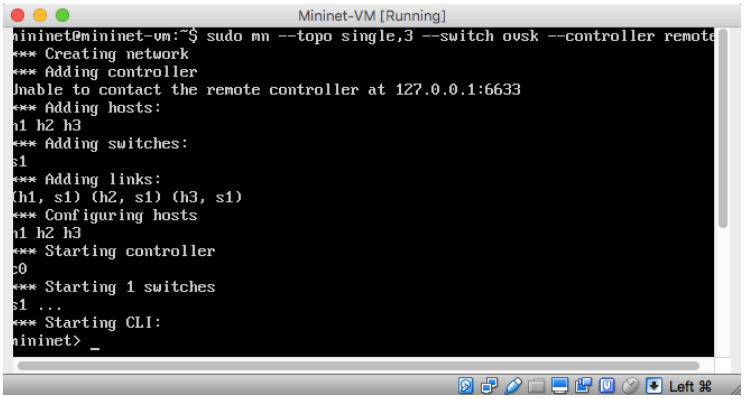


Figura 4.7 Creación de una topología con un controlador remoto

Al haber implementado una topología sin ningún tipo de controlador (no se ha ejecutado ningún controlador remoto), la tabla de flujos del switch debería estar vacía.. Se ejecuta el comando dpctl, para poder ver la tabla de flujo del switch (para comunicarse con el switch OpenFlow se envían mensajes al puerto 6634).

```
mininet@mininet-vm:~$dpctl dump-flows tcp:127.0.0.1:6634
```

Figura 4.8 Comprobación del contenido de la tabla de flujos

En la figura 4.8 se observa que no hay flujos instalados en la tabla de flujos del switch, esto hace que la conectividad sea nula, como se puede ver en la figura 4.9.

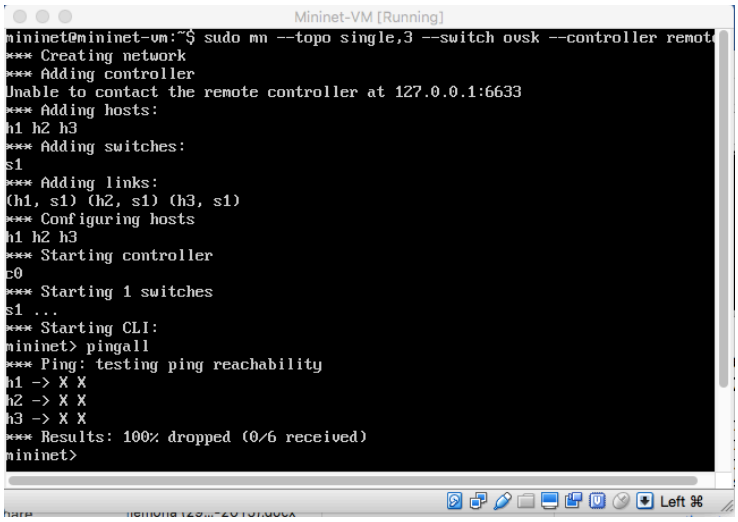


Figura 4.9 Comprobación de conectividad en la red

A continuación se procede a instalar manualmente los flujos necesarios para esta comunicación entre los hosts h1 y h2. Para ello se ejecutan los siguientes comandos:

mininet@mininet-vm:~$ dpctl add-flow tcp:127.0.0.1:6634 in_port=1,actions=output:2

mininet@mininet-vm:~$ dpctl add-flow tcp:127.0.0.1:6634 in_port=2,actions=output:1

Se vuelve a comprobar el contenido de la tabla de flujos y la conectividad de la red.

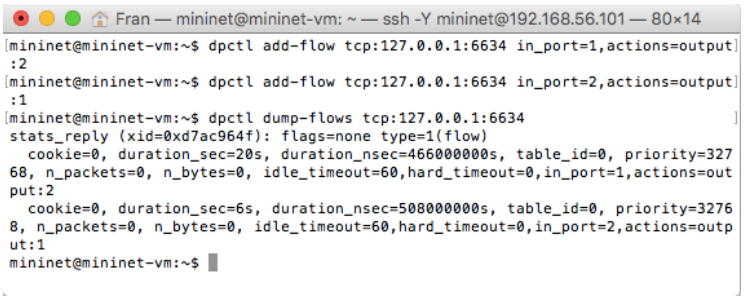


Figura 4.10 Comprobación del contenido de la tabla de flujos

Como se puede ver en la figura 4.10, ahora existen flujos instalados en la tabla de

flujos del switch. Y por lo tanto, existe conectividad entre los hosts h1 y h2, como se puede ver en la figura 4.11.

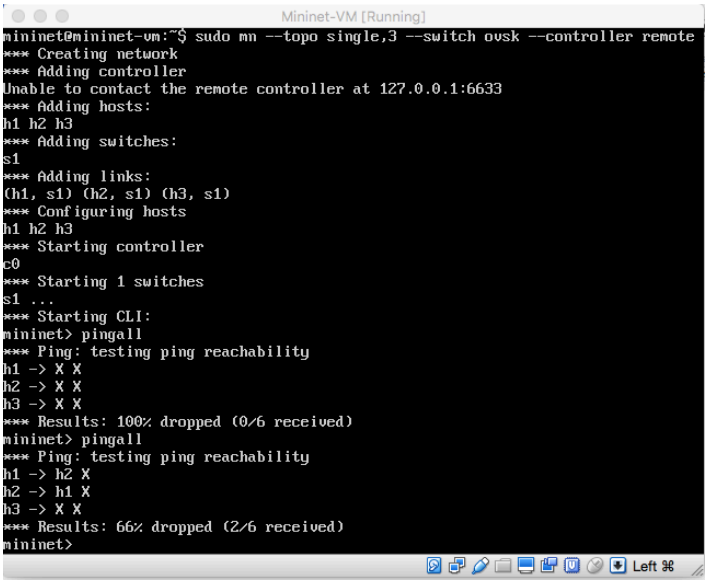


Figura 4.11 Comprobación de conectividad en la red

Con este ejemplo se puede comprobar que se dispone de un switch controlado por software, que es la esencia de la tecnología SDN. Y esto es lo que llevará a cabo automáticamente el controlador, en función de las instrucciones que le proporcionen las aplicaciones o componentes que se implementen.
Como ya se ha dicho, Mininet permite la ejecución de controladores remotos, como por ejemplo NOX, POX, Beacon,… En este caso se utilizará un controlador POX.
POX es un controlador que viene incorporado en las últimas versiones de la máquina virtual Mininet. Es un controlador derivado de NOX y es una plataforma de código abierto especialmente pensada para la investigación y el desarrollo de controladores OpenFlow. Está desarrollado en phyton permitiendo la programación de componentes en este lenguaje. Estos componentes son los que dan funcionalidad al controlador y es donde se pueden añadir características que proporcionen "valor añadido" a la red. Forman la esencia de las redes SDN.
Para lanzar un controlador POX hay que ejecutar el siguiente comando desde una consola SSH [40].

```
mininet@mininet-vm:~$ cd pox
mininet@mininet-vm:~$ ./pox.py componente
```

“componente” es el nombre de un archivo python (.py) que contiene la implementación de un componente.
Hay componentes ya desarrollados que proporcionan un punto de partida a partir de los cuales crear componentes propios. Como por ejemplo la implementación de un hub, un switch L2, servidores DHCP,…
En principio, no hay límite, la finalidad es crear componentes que hagan que la red se comporte como se desee, sin estar sujeto a los estrictos RFC ni al firmware propietario de los fabricantes de equipos.

A continuación, a modo de ejemplo se crea una topología con un controlador remoto y posteriormente se lanza un controlador POX que ejecuta el componente l2_learning. Este componente viene ya creado con POX y hace que los switches OpenFlow se comporten como un tipo de switch de aprendizaje L2.
Primero se crea una topología con tres switches con un host conectado a cada switch.

Se seleccionan switches OpenFlow y un controlador remoto.

```
mininet@mininet-vm:~$ sudo mn --topo linear,3 --switch ovsk --controller remote
```

Después, desde otra sesión SSH se ejecuta el controlador POX con el componente l2_learning:

```
mininet@mininet-vm:~$ ./pox.py forwarding.l2_learning
```

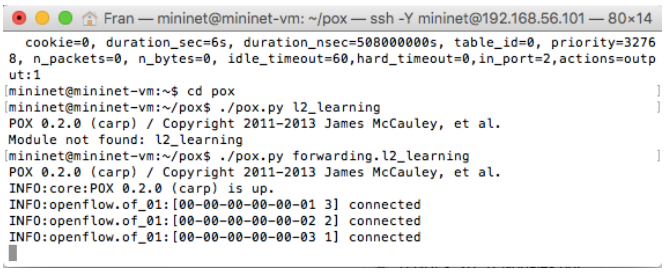


Figura 4.12 Ejecución del controlador POX

En la figura 4.12 se puede observar como se lanza el controlador POX y como este establece conexión con los tres switches existentes en la topología.
Finalmente, en la figura 4.13 se puede observar como existe conectividad entre los 3 host, lo que quiere decir que el controlador está funcionando correctamente.

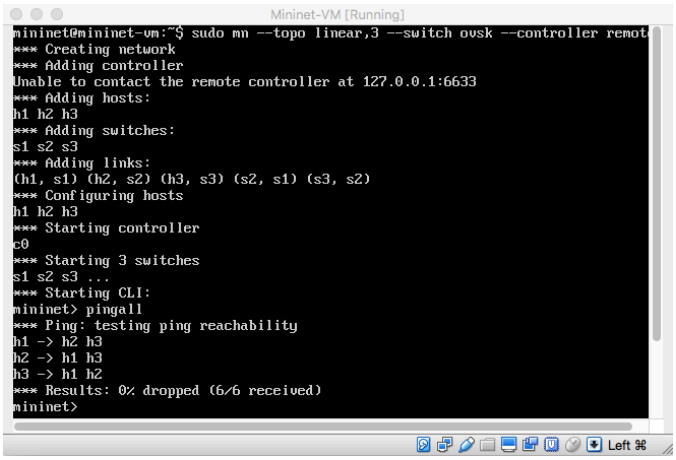


Figura 4.13 Comprobación de la conectividad

Una vez realizadas todas estas pruebas, se puede decir que el entorno SDN creado con Mininet-VM está operativo y funciona correctamente.
Por otro lado, también se observa que de las dos opciones de implementación de funcionalidades que se han visto (script utilizando la API de Mininet o implementación de un componente para un controlador remoto) la opción de la implementación de un componente para un controlador ofrece más posibilidades y permite aprovechar mejor las ventajas de las redes SDN. Ya que permite implementar componentes que hacen que la red se comporte como se desee. No obstante, los scripts de la API de Mininet son más sencillos de programar, incluso permiten la ejecución de comandos por la consola.
A pesar de esto, observando las pruebas realizadas y viendo las ventajas e inconvenientes de cada método, se opta por implementar los dos casos de uso mediante la utilización de ambos. Pudiendo ver así unos ejemplos prácticos de cada uno de ellos.

5. Capítulo 5, Implementación de un caso de uso.

5.1 Elección de las funcionalidades.

Como ejemplo de lo que se puede hacer a la hora de programar una red SDN, se implementarán dos casos de uso, ambos muy relacionados. Concretamente consistirán en dos simuladores de pérdidas de paquetes.
El primero consistirá en simular la pérdida del 4 % de los paquetes que pasan por el controlador. Para cada IP de origen, aplicará una pérdida de paquetes de un 4%.
El segundo es una "evolución" del caso anterior, ya que en este caso se eliminan sólo los paquetes de una determinada IP de origen y no de todas. Al igual que en el caso anterior también se eliminan el 4% de los paquetes, pero sólo de la IP indicada.

Estos ejemplos podrían servir para simular el comportamiento de ciertas aplicaciones en entornos con pérdidas de paquetes (VozIP, video por streaming,…).
Además, muestran lo fácilmente que se puede programar el tráfico de una red SDN y las posibilidades de programación que ofrecen estos tipos de redes.

En cuanto a la topología, para estas funcionalidades no se necesita ninguna topología especial, basta con la topología mínima de Mininet:

- 1 switch openFlow.
- 2 hosts conectados al switch.
- 1 controlador SDN conectado al switch.

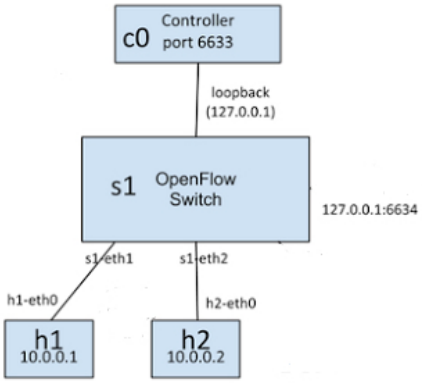


Figura 5.1 Diagrama de la topología que se utilizará

No obstante, se podría utilizar cualquier otra topología, el único requisito es que haya al menos dos host para poder intercambiarse paquetes. Y por supuesto, al menos un switch openFlow y un controlador.

5.2 Implementación de los casos de uso.

Caso de uso 1 (eliminar el 4% de los paquetes que provienen de cada IP diferente)

Esta funcionalidad se puede resolver con diferentes métodos:

- Crear una topología en la que se customizan los enlaces al crearlos. Se trata de fijar un 4% de pérdidas en los enlaces de los hosts con el switch en el momento que estos se crean.
- Implementar un componente para el controlador POX que compruebe los paquetes de entrada y elimine el 4% de los provenientes de cada IP (1 paquete de cada 25).

En ambos casos las pruebas han consistido en la realización de pings entre el host1 y el host2 de la topología creada y la posterior comprobación de los paquetes que se han perdido. Tras la ejecución de cada una de las series de pings se ha eliminado la topología, se ha vuelto a crear y se ha reiniciado el controlador para que todas las ejecuciones se realicen en igualdad de condiciones. El comando utilizado para la realización de los pings ha sido el siguiente:

mininet>h1 ping -c*XXX* h2

El parámetro -c seguido de un número indica el número de pings que se han de realizar (25, 26, 27, 50,…).

Solución 1

La solución más óptima es la primera, ya que no necesita de la implementación de ningún componente. Simplemente, al crear la topología, hay que indicar el tanto por ciento de pérdidas que se desea mediante el parámetro --link tc (del inglés Trafic Control). Este parámetro también permite fijar otras propiedades del enlace como el ancho de banda o el retardo.
Como ya se ha dicho anteriormente, la topología está formada por 1 switch openFlow, con 2 hosts conectados al switch y un controlador SDN conectado también al switch.
Para crear esta topología y definir la pérdida del 4% de paquetes en los enlaces, se ejecuta el siguiente comando:

mininet@mininet-vm:~$ sudo mn --switch ovsk --link tc, loss=4

Esto crea la topología mínima de Mininet, utilizando un switch openFlow y fijando un 4% de pérdidas en los enlaces. En la figura 5.2 se puede observar la creación de la topología deseada con la pérdida indicada.

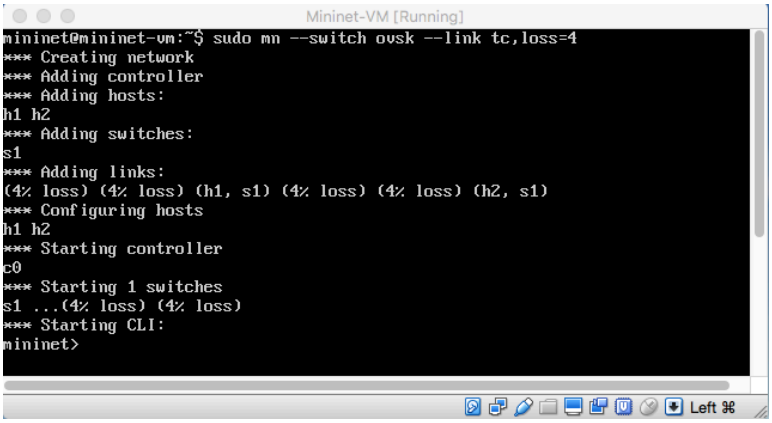


Figura 5.2 Creación de la topología con las pérdidas en los enlaces

Pruebas

Ahora ya se tiene la topología creada con una pérdida del 4% fijada en todos los enlaces. Por lo tanto ya se puede proceder a realizar pruebas para comprobar que esta topología se comporta como deseamos, es decir que se produzcan un 4% de pérdidas.
En la tabla 5.1 se pueden observar los resultados de las estadísticas ofrecidas por las diferentes ejecuciones del comando ping.

Paquetes enviados	Paquetes recibidos	Paquetes perdidos	% pérdidas
25	**20**	**5**	**20**
26	21	5	19
27	26	1	3
50	**41**	**9**	**18**
51	46	5	9
52	42	10	19
75	**64**	**9**	**14**
76	70	6	7
77	68	9	11
100	**82**	**18**	**18**
250	**211**	**39**	**15**
500	**425**	**75**	**15**
750	**645**	**115**	**14**
1000	**855**	**145**	**14**
1250	**1065**	**185**	**14**
1500	**1281**	**219**	**14**

Tabla 5.1 Resultados de los pings realizados del host1 al host2 (sol.1)

Como se puede comprobar en la tabla 5.1, los resultados de los pings muestran que efectivamente se produce una pérdida de paquetes. Pero también se observa que el comportamiento de la red, en cuanto a esta pérdida de paquetes, no sigue el patrón establecido que se esperaba. Especialmente en las series de pings pequeñas, donde se observa una gran variación en los resultados (tanto de paquetes perdidos como del tanto por ciento). Es más, tras la ejecución de secuencias de pings iguales, se han obtenido resultados algo distintos. En cambio, en las series grandes (más de 500) sí se observa una continuidad en las perdidas, concretamente se produce siempre una pérdida del 14% de los paquetes.
Esto hace pensar que esta solución, que parecía la más óptima, no es cien por cien adecuada ya que parece que se produce una pérdida superior al 4% (los enlaces eliminan los paquetes en ambos sentidos, envío y recepción). En el anexo 1 se puede consultar una tabla que muestra como se comportan los enlaces y las pérdidas que se producen.
Así que se procederá a implementar la segunda solución, menos óptima que esta pero que quizá permita afinar más, especialmente en las series de pocos pings.

Solución 2
La segunda solución consiste en la implementación de un componente en python para un controlador POX. Este componente eliminará el 4% de los paquetes (uno de cada 25) que provengan de la misma IP de origen.
La topología es la misma que en la solución 1 y está formada por 1 switch openFlow, con 2 hosts conectados al switch y un controlador SDN conectado también al switch, para enviarle las reglas y crear patrones de flujos de datos. La conexión entre el switch y el controlador se realiza de forma automática (sin necesidad de ser definida), ya que ambos se ejecutan en la misma máquina (en este caso virtual). Pero en este caso el controlador será un controlador POX remoto.
Para crear esta topología se ejecuta el siguiente comando:

```
mininet@mininet-vm:~$ sudo mn --switch ovsk –controller remote
```

Esto crea la topología deseada utilizando un switch openFlow e indicando que se

utilizará un controlador remoto.

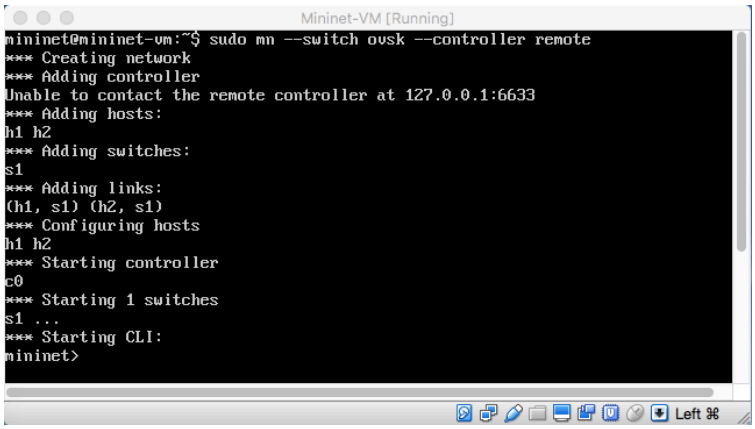


Figura 5.3 Creación de la topología con un controlador remoto

Tal y como se puede observar en la figura 5.3, ahora ya se tiene la topología creada y se puede lazar el controlador POX que ejecute el componente que se ha implementado. Este controlador será el encargado de eliminar los paquetes necesarios y enviarle al switch las reglas para crear patrones de flujos de datos. En el anexo 2 se muestra el código de este script (of_tutorialLoss.py).
Este script se ha implementado en python a partir del script of_tutorial.py, es decir, se ha cogido este script y se ha añadido el código necesario para que realice la funcionalidad deseada. En este caso la pérdida de un 4% de los paquetes enviados desde cada IP diferente. Concretamente, elimina para cada IP de origen un paquete de cada 25.
Una vez creado el componente se ejecuta el controlador POX con el siguiente comando:

mininet@mininet-vm:~/pox$ sudo python pox.py misc.of_tutorialLoss

Esto lanza un controlador POX que ejecuta el componente of_tutorialLoss.py. En la figura 5.4 se puede observar la ejecución del controlador y como este establece conexión con el switch.

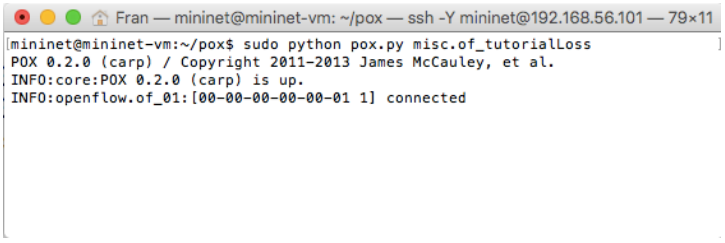


Figura 5.4 Ejecución del controlador POX y conexión con el switch

Pruebas
Llegados a este punto ya se dispone de un entorno SDN compuesto por la topología deseada y un controlador POX que ejecuta el componente of_tutorialLoss.py que se ha implementado. Por lo tanto ya se puede proceder a realizar pruebas para comprobar que este componente funciona correctamente. Al igual que en el caso anterior, estas pruebas han consistido en la realización de pings entre el host1 y el host2 de la topología creada y la posterior comprobación de los paquetes que se han perdido.
En la tabla 5.2 se pueden observar los resultados de las estadísticas ofrecidas por las diferentes ejecuciones del comando ping.

Paquetes enviados	Paquetes recibidos	Paquetes perdidos	% pérdidas
25	**24**	**1**	**4**
26	24	2	7
27	25	2	7
50	**47**	**3**	**6**
51	48	3	5
52	48	4	7
75	**70**	**5**	**6**
76	71	5	6
77	72	5	6
100	**93**	**7**	**7**
250	**231**	**19**	**7**
500	**461**	**39**	**7**
750	**692**	**58**	**7**
1000	**922**	**78**	**7**
1250	**1152**	**98**	**7**
1500	**1383**	**117**	**7**

Tabla 5.2 Resultados de los pings realizados del host1 al host2 (sol.2)

Como se puede comprobar en la tabla 5.2, los resultados de los pings muestran que efectivamente se produce una pérdida de paquetes. Pero también se observa que el comportamiento de la red, en cuanto al tanto por ciento de pérdida, no sigue el patrón establecido que se esperaba. Especialmente para series de pings pequeñas, donde se observa una cierta variación en los resultados. En cambio, en las series grandes (más de 100) sí se observa una continuidad en las perdidas, concretamente se produce siempre una pérdida del 7% de los paquetes.
Esto podría hacer pensar que esta solución, que parecía más afinada que la anterior, no es cien por cien adecuada para implementar este caso de uso. Pero no es así, si se analizan los resultados con detalle se observa que sí son correctos.
A la hora de analizar los resultados hay que tener en cuenta ciertas cosas:

- Para cada ping, por el controlador pasan 2 paquetes, la petición del host1 y la respuesta del host2. Por lo tanto todo hace pensar que por cada 25 pings (50 paquetes) se han de eliminar 2 (un 4%), pero no es del todo cierto.
- El host1 envía todos los paquetes y es el controlador quien decide si ese paquete se envía realmente o se bloquea. Por lo tanto, de cara a la estadística que muestra el ping al finalizar, para el host1 todos sus paquetes se han enviado (aunque algunos realmente hayan sido bloqueados por el controlador).
- Cuando una petición se bloquea, como ya se ha dicho, no cuenta como perdido, pero al no enviarse realmente al host2 tampoco se recibe ninguna respuesta. Por lo tanto, en este caso sí que cuenta como paquete perdido, ya que el host1 espera la respuesta.
- Para comprobar los resultados de esta solución hay que fijarse en los paquetes perdidos más que en el tanto por ciento.

Con un ejemplo se verá más claro. Por ejemplo en el caso de 25 pings aparece un único paquete como perdido y no 2. Esto se debe a que la petición 25 del host1 se bloquea, pero no cuenta como pérdida ya que el host1 no sabe que está bloqueada. Pero como al host2 no llega ninguna petición, este no envía ninguna respuesta, pero el host1 sí que espera una. Y esta respuesta que espera y no llega es el paquete perdido que aparece en la estadística. Por esto aparece un único paquete perdido en

lugar de 2. A partir de aquí, la cosa se complica un poco.
Como el host2 no ha recibido la petición 25 del host1, no ha contestado. Por lo tanto, el host1 ha enviado 25 paquetes mientras que el host2 ha enviado 24. Esto hace que en la siguiente petición del host1 (la 26) la respuesta del host2 (la 25) se bloquee. Debido a esto, se produce un desfase entre el número de paquetes enviados por el host1 y los enviados por el host2.
Dicho de otro modo, para una serie de por ejemplo 100 pings tendríamos:

- 100 paquetes enviados por el host 1, de los cuales se eliminan el 4%
- Esto hace que al host 2 lleguen únicamente 96 paquetes.
- El host 2 envía 96 paquetes de respuesta, pero de estos 96 se filtran el 4% (3.84 paquetes ≅ 3).
- Con lo cual al host 1 llegan 93 paquetes, que es lo que aparece en la estadística del ping.

Este cálculo se puede repetir para el resto de series de pings y comprobar que concuerdan con los resultados de las estadísticas de los pings.
En la tabla del anexo 3 se muestra de forma detallada la evolución de ambos contadores y los bloqueos que se producen mientras se van realizando las diferentes peticiones de ping. Comparando esta tabla (evolución teórica del componente implementado) con la tabla 5.2 (resultados reales), se observa que los datos de ambas tablas coinciden y por lo tanto los datos teóricos son correctos. Además, se puede comprobar como cada 25 paquetes enviados por un host se bloquea 1 (un 4%).
También se puede observar en la tabla 5.3 como los resultados teóricos concuerdan con los resultados reales obtenidos en los pings (a la hora de eliminar paquetes sólo se tiene en cuenta la parte entera, no se puede eliminar una parte de un paquete).

Salen de h1	Controlador elimina 4% de los que salen de h1	Llegan a h2 y contesta	Controlador elimina 4% de los que salen de h2	Llegan a h1	Resultados PINGs	Diferencia
25	**1**	**24**	**0,96 ≈ 0**	**24**	**24**	**0**
26	1,04 ≈ 1	25	1	24	24	0
27	1,08 ≈ 1	26	1,04 ≈ 1	25	25	0
50	**2**	**48**	**1,92 ≈ 1**	**47**	**47**	**0**
51	2,04 ≈ 2	49	1,96 ≈ 1	48	48	0
52	2,08 ≈ 2	50	2	48	48	0
75	**3**	**72**	**2,88 ≈ 2**	**70**	**70**	**0**
76	3,04 ≈ 3	73	2,92 ≈ 2	71	71	0
77	3,08 ≈ 3	74	2,96 ≈2	72	72	0
100	**4**	**96**	**3,84 ≈ 3**	**93**	**93**	**0**
250	**10**	**240**	**9,6 ≈ 9**	**231**	**231**	**0**
500	**20**	**480**	**19,2 ≈ 19**	**461**	**461**	**0**
750	**30**	**720**	**28,8 ≈ 28**	**692**	**692**	**0**
1000	**40**	**960**	**38,4 ≈ 38**	**922**	**922**	**0**
1250	**50**	**1200**	**48**	**1152**	**1152**	**0**
1500	**60**	**1440**	**57,6 ≈ 57**	**1383**	**1383**	**0**

Tabla 5.3 Comparativa resultados teóricos y resultados reales (sol.2)

Dicho esto, se puede afirmar que el controlador que incorpora el componente que se ha implementado (of_tutorialLoss.py), funciona correctamente y cumple con las premisas deseadas (eliminar 1 paquete de cada 25 para cada IP de origen).

Caso de uso 2 (eliminar el 4% de los paquetes que provienen de una determinada IP)

Al igual que en el caso de uso anterior, esta funcionalidad se puede resolver con diferentes métodos:

- Crear una topología en la que se fija un 4% de pérdidas en la interface de un host con una IP determinada.
- Implementar un componente para el controlador POX que compruebe los paquetes de entrada y elimine el 4% de los provenientes de una determinada IP.

Solución 1

La solución más óptima es la primera, ya que no necesita de la implementación de ningún controlador. Simplemente, es necesario crear la topología y fijar un tanto por ciento de pérdidas en la interface de conexión con el switch del host con la IP indicada. Para indicar el tanto por ciento de pérdidas que se desea se utiliza el comando tc (del inglés Trafic Control). Este parámetro también permite fijar otras propiedades del enlace como el ancho de banda o el retardo.
Como ya se ha dicho anteriormente, la topología está formada por 1 switch, con 2 hosts conectados al switch y un controlador SDN conectado también al switch, para enviarle las reglas y crear patrones de flujos de datos.
Para crear esta topología y definir la pérdida del 4% de paquetes en la interface del host con la IP indicada, se ha implementado el script “topotc.py” a partir del script “simpleperf.py”. En el anexo 4 se muestra el código de este script (topotc.py).
Para ejecutar este script hay que ejecutar el siguiente comando:

```
mininet@mininet-vm:~/mininet/examples$ sudo ./topotc.py 10.0.0.1
```

Esto crea la topología deseada, fijando un 4% de pérdidas en la interface del host con la IP indicada como parámetro. En la figura 5.5 se puede observar la creación de la topología deseada con la pérdida en la IP deseada.

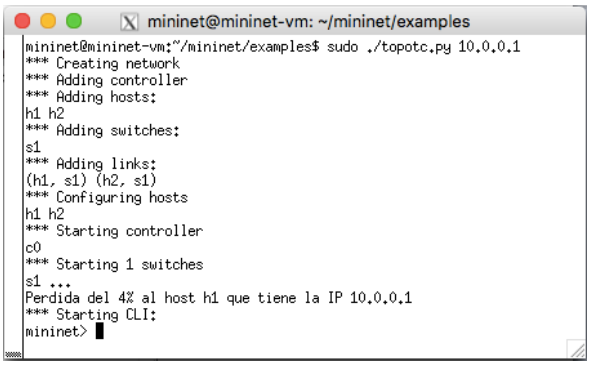


Figura 5.5 Ejecución del script topotc.py

Pruebas

Ahora ya se tiene la topología creada con una pérdida del 4% fijada la interface del host con la IP indicada como parámetro. Por lo tanto ya se puede proceder a realizar pruebas para comprobar que esta topología se comporta como deseamos, es decir que se produzcan un 4% de pérdidas en los paquetes que envía el host con la IP indicada.
En la tabla 5.4 se pueden observar los resultados de las estadísticas ofrecidas por las diferentes ejecuciones del comando ping.

Paquetes enviados	Paquetes recibidos	Paquetes perdidos	% pérdidas
25	**24**	**1**	**4**
26	23	3	11
27	27	0	0
50	**46**	**4**	**8**
51	50	1	1
52	49	3	5
75	**74**	**1**	**1**
76	75	1	1
77	74	3	3
100	**97**	**3**	**3**
250	**235**	**15**	**6**
500	**481**	**19**	**3**
750	**725**	**25**	**3**
1000	**957**	**43**	**4**
1250	**1206**	**44**	**3**
1500	**1440**	**60**	**4**

Tabla 5.4 Resultados de los pings realizados del host1 al host2 (sol.1)

Como se puede comprobar en la tabla 5.4, los resultados de los pings muestran que efectivamente se produce una pérdida de paquetes. Pero también se observa que el comportamiento de la red, en cuanto a esta pérdida de paquetes, no sigue el patrón establecido que se esperaba. Especialmente en las series de pings pequeñas, donde se observa una gran variación en los resultados (tanto de paquetes perdidos como del tanto por ciento). Es más, tras la ejecución de secuencias de pings iguales, se han obtenido resultados algo distintos. En cambio, en las series grandes (más de 500) sí se observa una cierta continuidad en las perdidas, concretamente se produce una pérdida que oscila entre el 3 y el 4 por ciento de los paquetes.
Esto hace pensar que esta solución, que parecía la más óptima, no es cien por cien adecuada ya que no es muy regular. En el anexo 5 se puede consultar una tabla que muestra como se comportan los enlaces y las pérdidas que se producen.
Así que se procederá a implementar la segunda solución, menos óptima que esta pero que quizá permita afinar más y que de unos resultados más regulares.

Solución 2

La segunda solución consiste en la implementación de un componente en python para un controlador POX. Este componente eliminará el 4% de los paquetes (uno de cada 25) que provengan de una determinada IP.
La topología es la misma que en la solución 1 y está formada por 1 switch openFlow, con 2 hosts conectados al switch y un controlador SDN conectado también al switch, para enviarle las reglas y crear patrones de flujos de datos. La conexión entre el switch y el controlador se realiza de forma automática (sin necesidad de ser definida), ya que ambos se ejecutan en la misma máquina (en este caso virtual). Pero en este caso el controlador será un controlador POX remoto.
Para crear esta topología se ejecuta el siguiente comando:

```
mininet@mininet-vm:~$ sudo mn --switch ovsk --controller remote
```

Esto crea la topología deseada utilizando un switch openFlow e indicando que se utilizará un controlador remoto.

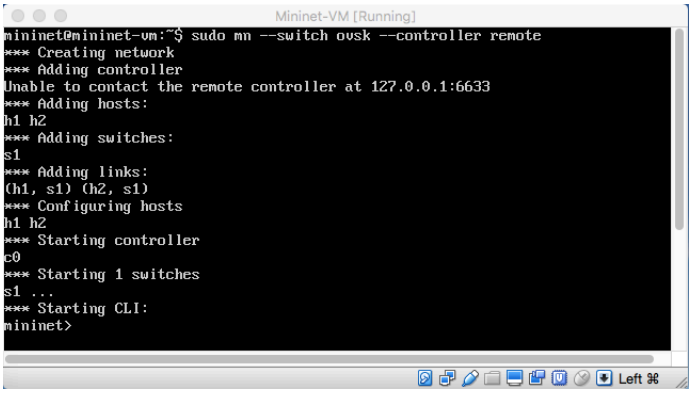


Figura 5.6 Creación de la topología con un controlador remoto

Ahora ya se tiene la topología creada y se puede lazar el controlador POX que ejecute el componente que se ha implementado. Este controlador será el encargado de eliminar los paquetes necesarios y enviarle al switch las reglas para crear patrones de flujos de datos. En el anexo 6 se muestra el código de este script (of_tutorialLoss2.py). Este script se ha implementado en python a partir del script of_tutorialLoss.py implementado en el caso de uso 1, es decir, se ha cogido este script y se ha añadido el código necesario para recoger por parámetro la IP deseada. Concretamente, este componente elimina 1 paquete de cada 25 que envía la IP que se indica por parámetro.
Una vez creado el componente se ejecuta el controlador POX con el siguiente comando:

```
mininet@mininet-vm:~/pox$ sudo python pox.py misc.of_tutorialLoss2 --IP=10.0.0.1
```

Esto lanza un controlador POX que ejecuta el componente of_tutorialLoss2.py. El parámetro --IP sirve para indicar la IP a la que se quiere aplicar la pérdida de paquetes. En la figura 5.7 se puede observar la ejecución del controlador y como este establece conexión con el switch.

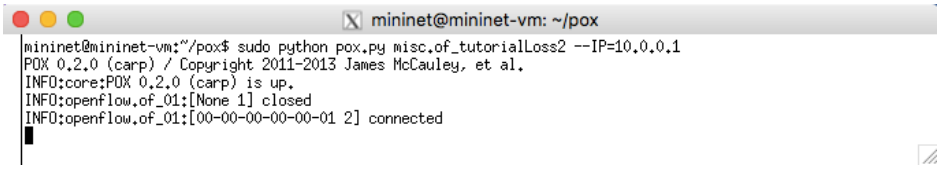


Figura 5.7 Ejecución del controlador POX y conexión con el switch

Pruebas
Llegados a este punto ya se dispone de un entorno SDN compuesto por la topología deseada y un controlador POX que ejecuta el componente of_tutorialLoss2.py que se ha implementado. Por lo tanto ya se puede proceder a realizar pruebas para comprobar que este componente funciona correctamente. Al igual que en los casos anteriores, estas pruebas han consistido en la realización de pings entre el host1 y el host2 de la topología creada y la posterior comprobación de los paquetes que se han perdido.
En la tabla 5.5 se pueden observar los resultados de las estadísticas ofrecidas por las diferentes ejecuciones del comando ping.

Paquetes enviados	Paquetes recibidos	Paquetes perdidos	% pérdidas
25	**24**	**1**	**4**
26	25	1	3
27	26	1	3
50	**48**	**2**	**4**
51	49	2	3
52	50	2	3
75	**72**	**3**	**4**
76	73	3	3
77	74	3	3
100	**96**	**4**	**4**
250	**240**	**10**	**4**
500	**480**	**20**	**4**
750	**720**	**30**	**4**
1000	**960**	**40**	**4**
1250	**1200**	**50**	**4**
1500	**1440**	**60**	**4**

Tabla 5.5 Resultados de los pings realizados del host1 al host2 (sol.2)

Como se puede comprobar en la tabla 5.5, los resultados de los pings muestran que efectivamente se produce una pérdida de paquetes. Pero también se observa que el comportamiento de la red, en cuanto al tanto por ciento de pérdida, no sigue exactamente el patrón establecido que se esperaba ya que oscila entre el 3% y el 4%. Esto se debe a que, como ya se ha dicho anteriormente, se eliminan 1 de cada 25 paquetes.
Por lo tanto, tal y como se puede comprobar en la tabla, el comportamiento es totalmente correcto, ya que en todas las series de pings múltiples de 25 se produce un 4% de pérdidas (independientemente de si son series de muchos pings o pocos).
Dicho esto, se puede afirmar que el controlador que incorpora el componente que se ha implementado (of_tutorialLoss2.py), funciona correctamente y cumple con las premisas deseadas (eliminar 1 paquete de cada 25 para una determinada IP).

Resumen
Viendo los resultados de ambos casos de usos, se puede decir que se han conseguido los objetivos que se propusieron al principio.
Pero además de esto, se puede llegar a la conclusión que las redes SDN permiten un gran control del tráfico de la red y de una forma muy centralizada. Ya sea con comandos a través de línea de comandos (soluciones 1 de cada caso de uso) o implementando componentes para los controladores (soluciones 2 de cada caso de uso).

En este caso se han implementado dos variantes de simuladores de pérdidas de paquetes, pero en lugar de esto se podría haber implementado cualquier otra funcionalidad, como por ejemplo un firewall que elimine todos los paquetes de ciertas IPs de origen, o de destino. O un control de acceso que sólo permitiera el tráfico de ciertas MACs,…

Como se puede comprobar las posibles implementaciones son muchas.

6. Capítulo 6, Valoración productiva y presupuestaria.

6.1 Valoración productiva

Para la realización de este proyecto se han utilizado varios roles:

- **Consultor.**
  Las horas imputadas al consultor han consistido mayormente en el estudio y análisis de las tecnologías SDN y NFV, los protocolos y proyectos relacionados con estas tecnologías y los distintos productos y empresas existentes en el mercado. Básicamente los capítulos 2 y 3 de esta memoria, que han supuesto un total de 79 horas.
- **Jefe de proyecto.**
  Las horas imputadas al jefe de proyecto han consistido mayormente en la realización de la planificación, coordinación entre los diferentes roles, seguimiento del proyecto con el director, hacer la valoración económica, las conclusiones,… que han supuesto un total de 20 horas.
- **Analista-programador.**
  Las horas imputadas al analista-programador han consistido mayormente en la creación del entorno SDN y la implementación de los 2 casos de uso. Básicamente los capítulos 4 y 5 , que han supuesto un total de 113 horas.

Además de estas horas imputadas a estos roles, se han destinado 15 horas a la realización del resto de capítulos de la memoria y a su revisión final.

6.2 Valoración presupuestaria

Para realizar el cálculo de los costes del proyecto, se han tenido en cuenta 2 tipos de gastos:

- **Personales.**
  Correspondientes a las horas de trabajo realizadas por los diferentes roles.
- **Materiales.**
  Correspondientes al software y hardware utilizado.

Gastos personales

A continuación se muestra una tabla con las horas imputadas a los diferentes roles (explicado en el capítulo 6.1) y su valoración económica.

Rol	Horas	Precio hora	Coste
Consultor	79	50 €	3.950 €
Jefe de proyecto	20	60 €	1.200 €
Analista-programador	113	35 €	3.955 €
TOTAL gastos personales			9.105 €

Tabla 6.1 Costes personales

Gastos materiales

Los gastos materiales correspondientes al proyecto se pueden dividir en dos tipos:

- **Licencias de software.**
  Concretamente se ha utilizado un paquete de Office 2013 y el Project 2013.
  El resto de software ha sido software libre.
- **Hardware.**
  Únicamente se ha utilizado un portátil con conexión a internet.

Los costes de estos materiales han sido los siguientes:

Software	Precio
Microsoft Office hogar y estudiantes 2013	116,95 €
Microsoft Project standard 2016	769 €
Portátil	1.000 €
TOTAL software	1.885,95 €

Tabla 6.2 Costes materiales

Gastos Totales

Finalmente los gastos totales del proyecto se obtienen de sumar los gastos personales y los gastos materiales.

Gasto	Precio
Gastos materiales	1.885,95 €
Gastos personales	9.105 €
TOTAL coste proyecto	**10.990,95 €**

Tabla 6.3 Costes totales del proyecto

7. Capítulo 7, Conclusiones.

El principal objetivo que se planteó en este proyecto era estudiar y conocer las tecnologías SDN y NFV. Con la vista puesta en este objetivo, se ha realizado una amplia investigación de estas tecnologías, así como de protocolos, empresas, productos,.. relacionados con ellas. Una vez finalizado este estudio, se puede decir que estas tecnologías ofrecen un gran abanico de posibilidades gracias a que alejan las redes del hardware y las acercan al software. El hecho de que SDN y NFV permitan cambiar los equipos de red actuales (con un comportamiento predefinido por el fabricante) por equipos que permiten ser programados según las necesidades del administrador, hace que las aplicaciones prácticas de estas tecnologías sean innumerables. No obstante, estas tecnologías también presentan algunos inconvenientes y además son aún "jóvenes". Esto me hace pensar que ambas tienen aún un largo camino por delante antes de que puedan sustituir completamente a las redes actuales. A pesar de esto, creo que en los próximos años se verá cómo se van desplegando algunas SDN/NFV con nuevas funcionalidades (como ya han hecho Google y Microsoft).
Dado que al comenzar este proyecto desconocía completamente estas tecnologías, este estudio ha sido para mí una experiencia muy enriquecedora.

Al comenzar la parte práctica de este proyecto, se buscaron sistemas que permitieran crear un entorno SDN y programar y controlar los flujos de datos de la red. Tras contemplar diferentes opciones, se decidió utilizar el simulador de redes Mininet y un controlador SDN POX. POX está indicado para el desarrollo de componentes y además funciona perfectamente con Mininet, uno de los simuladores SDN más utilizado. En los ejemplos prácticos realizados en este proyecto (eliminación del 4% de paquetes para cada IP origen y del 4 % para una determinada IP), se ha creado una topología con 2 hosts conectados a 1 switch y un controlador conectado al switch. Para cada caso de uso se han aportado 2 soluciones, una utilizando la API de mininet y otra un controlador POX (que incorpora los componentes implementados). Una vez hechas las pruebas, se comprueba que se han conseguido cumplir las premisas iniciales.
Viendo los resultados de ambos casos de uso, llego a la conclusión que las redes SDN permiten un gran control del tráfico de la red y de una forma muy centralizada. Ya sea con comandos a través de línea de comandos o implementando componentes para los controladores.
Personalmente esta última parte ha presentado una especial dificultad, ya que mi desconocimiento del lenguaje de programación (python) y de la API, ha hecho que la implementación se haya alargado más de lo previsto.

Finalmente, debido a mi total desconocimiento de las tecnologías SDN y NFV, he tenido la oportunidad de afrontar un desafío personal que me ha llevado a conocer estas tecnologías, tanto desde un punto de vista teórico como práctico.
Todo esto me hace pensar que estudios como el realizado en este proyecto contribuyen a la divulgación de estas tecnologías.

Después de estudiar las tecnologías SDN y NFV, creo que se puede asegurar que estas tecnologías son el futuro de las redes, ya que ambas permiten aprovechar mucho mejor los recursos que las tecnologías actuales. De hecho, muchas empresas empiezan a confiar cada día más en estas nuevas tecnologías. Dicho esto, también creo que estas nuevas redes no sustituirán a las redes actuales a corto plazo, más bien pienso que coexistirán aprovechando lo mejor de cada una de ellas. No obstante, es inevitable pensar que en el futuro las redes sufrirán una profunda transformación posiblemente llegando a unos límites que no se contemplan actualmente.

Glosario

API (Application Programming Interface): Interfaz de Programación de Aplicaciones. Es el conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

CAPEX (Capital Expenditure): Dinero que invierte una empresa en la mejora o adquisición de productos o servicios.

CPD (Centro de procesamiento de datos): Lugar en el que se almacenan todos los recursos necesarios para el procesamiento de la información de una empresa.

DC (Data Center): Lugar en el que se almacenan los recursos físicos y lógicos necesarios para el procesamiento y control de la información de una empresa.

DoS, Ataque (Denial of Service): Ataque de Denegación De Servicio. Es un ataque a un sistema de computadoras o red que causa que un servicio o recurso sea inaccesible a los usuarios legítimos, debido a una sobrecarga de los recursos.

IDS (Intrusion Detection System): Sistema de Detección de Intrusiones. Es un programa de detección de accesos no autorizados a un ordenador o a una red.

IoT (Internet of Things): Objetos de la vida cotidiana que se conectan a internet: relojes, TV,…

IP (Internet Protocol): Protocolo de red para el envío de paquetes de datos tanto a nivel local como a través de redes.

ISP (Internet Service Provider): Proveedor de Servicios de Internet. Un ISP conecta a sus usuarios a Internet.

LAN (Local Area Network): Red de Área Local. Es una red de computadoras que abarca un área reducida a una casa, un departamento o un edificio.

NAT (Network Address Translation): Traducción de Direcciones de Red. Es un mecanismo utilizado por Routers IP para intercambiar paquetes entre dos redes que asignan mutuamente direcciones incompatibles.

NFV (Network Functions Virtualization): Virtualización de Funciones de Red. Arquitectura de red que utiliza técnicas relacionadas con la virtualización de los servicios TI.

OPEX (Operational Expenditure): Dinero que invierte una empresa en el mantenimiento de los productos o servicios.

Overlay (red Overlay): Red lógica creada encima de una red física que ya existente.

PaaS (Platform as a Service): Plataforma como Servicio. Es la encapsulación de una abstracción de un ambiente de desarrollo y el empaquetamiento de una serie de módulos o complementos que proporcionan, normalmente, una funcionalidad horizontal (persistencia de datos, autenticación, mensajería, etc.).

SDN (Software Defined Networking): Red Definida por Software. Tecnología que permite administrar los servicios de red mediante la abstracción del plano de datos.

TCP/IP (Transmission Control Protocol/Internet Protocol): Conjunto de protocolos de red en los que se basa Internet para la transmisión de datos entre ordenadores.

TELNET (Teletype Network): Protocolo de red para acceder remotamente a otro equipo.

VLAN (Virtual LAN): Método para crear redes lógicas independientes de la red física.

VM (Virtual Machine): Máquina Virtual. Software que simula a un host y que puede ejecutar programas como si fuese un ordenador real.

VPN (Virtual Private Network): Red Privada Virtual. Es una tecnología de red que permite una extensión segura de la red local (LAN) sobre una red pública o no controlada como Internet

VXLAN (Virtual Extensible LAN): Tecnología de virtualización de red para crear redes de capa 2 y transportarlas de forma transparente por redes de capa 3.

WAN (Wide Area Network): Red que une varias redes LAN.

XML (Extensible Markup Language): Lenguaje que permite el almacenamiento y etiquetado de documentos web.

Bibliografía

[1] M Fos, Ana (27/05/2014) SDN y NFV protagonizan la transformación de las redes, Telecomunicaciones para gerentes. Fecha de consulta: 4 de Octubre de 2015. URL: http://www.telecomunicacionesparagerentes.com/sdn-y-nfv-protagonizan-la-transformacion-de-las-redes/

[2] Rodrigo, Juan (05/02/2015) SDN y NFV: Nuevo paradigma en las comunicaciones, Teldat. Fecha de consulta: 4 de Octubre de 2015. URL: http://blog.teldat.com/?p=325&lang=es

[3] Wikipedia (09/09/2015) Redes definidas por software, Wikipedia La enciclopedia libre. Fecha de consulta: 4 de Octubre de 2015. URL: https://es.wikipedia.org/wiki/Redes_definidas_por_software

[4] Rouse, Margaret (11/2012) Redes definidas por software (SDN): Definición, TechTarget. Fecha de consulta: 5 de Octubre de 2015. URL: http://searchdatacenter.techtarget.com/es/definicion/Redes-definidas-por-software-SDN

[5] Larsen DeCarlo, Amy (11/2013) ¿Pensando implementar SDN en su empresa? No tan rápido, TechTarget. Fecha de consulta: 6 de Octubre de 2015. URL: http://searchdatacenter.techtarget.com/es/consejo/Pensando-implementar-SDN-en-su-empresa-No-tan-rapido

[6] Wikipedia (17/08/2015) NFV, Wikipedia La enciclopedia libre. Fecha de consulta: 6 de Octubre de 2015. URL: https://es.wikipedia.org/wiki/NFV

[7] Wikipedia (21/09/2015) Network function virtualization, Wikipedia, the free encyclopedia. Fecha de consulta: 6 de Octubre de 2015. URL: https://en.wikipedia.org/wiki/Network_function_virtualization

[8] Lemke, Andreas (08/12/2014) Redes SDN: Llevar la funcionalidad NFV al siguiente nivel, Alcatel-Lucent. Fecha de consulta: 7 de Octubre de 2015. URL: https://techzine.alcatel-lucent.com/es/redes-sdn-llevar-la-funcionalidad-nfv-al-siguiente-nivel

[9] McNickle, Michelle (06/2014) Diez definiciones esenciales de virtualización de redes, TechTarget. Fecha de consulta: 7 de Octubre de 2015. URL: http://searchdatacenter.techtarget.com/es/consejo/Diez-definiciones-esenciales-de-virtualizacion-de-redes

[10] (28/07/2015) NFV y SDN como motores de la transformación digital, El blog de Interoute Iberia. Fecha de consulta: 7 de Octubre de 2015. URL: http://www.interoute.es/blog/nfv-y-sdn-motores-transformacion-digital

[11] Martínez, Fernanda (20/02/2015) NFV: ¿Qué es; cómo se usa y por qué debes saberlo?, CIO América Latina. Fecha de consulta: 7 de Octubre de 2015. URL: http://www.cioal.com/2015/02/20/nfv-que-es-como-se-usa-por-que-debes-saberlo

[12] M Fos, Ana (21/05/2013) SDN, hacia un nuevo concepto de infraestructura de red, Telecomunicaciones para gerentes. Fecha de consulta: 8 de Octubre de 2015. URL: http://www.telecomunicacionesparagerentes.com/sdn-hacia-un-nuevo-concepto-de-infraestructura-de-red

[13] Dell (2015) Strategy for Software-Defined Networking, Dell. Fecha de consulta: 8 de Octubre de 2015. URL: http://www.dell.com/learn/us/en/555/solutions/networking-strategy-for-software-defined-networking

[14] Maistre, Ray Le (23/10/2012) Google: SDN Works for Us, Light Reading. Fecha de consulta: 12 de Octubre de 2015. URL: http://www.lightreading.com/carrier-sdn/sdn-architectures/google-sdn-works-for-us/d/d-id/699197

[15] Cisco (2015) Cisco ONE Software, Cisco. Fecha de consulta: 12 de Octubre de 2015. URL: http://www.cisco.com/c/en/us/products/software/one-software/index.html#~overview

[16] HP (2015) Software-Defined Networking, an agile, automated, programmable network from data center to campus and branch, HP. Fecha de consulta: 12 de Octubre de 2015. URL: http://www8.hp.com/us/en/networking/sdn/index.html

[17] IBM (2015) Networking Services IBM Software Defined Networking, IBM. Fecha de consulta: 12 de Octubre de 2015. URL: http://www-935.ibm.com/services/us/en/it-services/networking-services/software-defined-network/index.html

[18] Duffy, Jim (18/07/2015) Google opens up on its SDN, Network World. Fecha de consulta: 12 de Octubre de 2015. URL: http://www.networkworld.com/article/2937656/cisco-subnet/google-opens-up-on-its-sdn.html

[19] Duffy, Jim (17/06/2015) Microsoft needs SDN for Azure cloud, Network World. Fecha de consulta: 12 de Octubre de 2015. URL: http://www.networkworld.com/article/2937396/cloud-computing/microsoft-needs-sdn-for-azure-cloud.html

[20] Wikipedia (28/05/2015) OpenFlow, Wikipedia La enciclopedia libre. Fecha de consulta: 13 de Octubre de 2015. URL: https://es.wikipedia.org/wiki/OpenFlow

[21] Wikipedia (09/09/2015) OpenStack, Wikipedia La enciclopedia libre. Fecha de consulta: 14 de Octubre de 2015. URL: https://es.wikipedia.org/wiki/OpenStack

[22] Wikipedia (31/08/2015) OpenDaylight Project, Wikipedia, the free encyclopedia. Fecha de consulta: 14 de Octubre de 2015. URL: https://en.wikipedia.org/wiki/OpenDaylight_Project

[23] McNickle, Michelle (09/2014) Cinco protocolos SDN que no son OpenFlow, TechTarget. Fecha de consulta: 15 de Octubre de 2015. URL: http://searchdatacenter.techtarget.com/es/cronica/Cinco-protocolos-SDN-que-no-son-OpenFlow

[24] McNickle, Michelle (10/2014) Cinco controladores SDN comerciales que hay que conocer, TechTarget. Fecha de consulta: 16 de Octubre de 2015. URL: http://searchdatacenter.techtarget.com/es/cronica/Cinco-controladores-SDN-comerciales-que-hay-que-conocer

[25] Pate, Prayson (30/03/2013) NFV and SDN: What's the Difference?, sdxcentral. Fecha de consulta: 16 de Octubre de 2015. URL: https://www.sdxcentral.com/articles/contributed/nfv-and-sdn-whats-the-difference/2013/03

[26] sdxcentral () Which is Better – SDN or NFV?, sdxcentral. Fecha de consulta: 16 de Octubre de 2015. URL: https://www.sdxcentral.com/resources/nfv/which-is-better-sdn-or-nfv

[27] Garrison, Steve (11/02/2014) Understanding the differences between Software Defined Networking, network virtualization and Network Functions Virtualization, Network World. Fecha de consulta: 16 de Octubre de 2015. URL: http://www.networkworld.com/article/2174268/tech-primers/understanding-the-differences-between-software-defined-networking-network-virtualizati.html

[28] Bartolik, Pete (11/08/2014) Confused by SDN & NFV?, Network World. Fecha de consulta: 17 de Octubre de 2015. URL: http://www.networkworld.com/article/2462477/software-defined-networks/confused-by-sdn-and-nfv.html

[29] Ciena (2015) Las tres formas clave en que NFV y SDN ponen a los operadores de redes en el asiento del conductor, Ciena: la especialista en redes. Fecha de consulta: 19 de Octubre de 2015. URL: http://www.ciena.com.mx/resources/quick-content/Three-Key-Ways-NFV-and-SDN-Put-Network-Operators-in-the-Drivers-Seat-QC-ENG_es_LA.html

[30] El economista (19/12/2014) COMUNICADO: Huawei considerada principal proveedor SDN y NFV por Current Analysis, El economista. Fecha de consulta: 19 de Octubre de 2015. URL: http://www.eleconomista.es/internacional/noticias/6342722/12/14/COMUNICADO-Huawei-considerada-principal-proveedor-SDN-y-NFV-por-Current-Analysis.html

[31] Mininet (2015) Download/Get Started With Mininet, Mininet. Fecha de consulta: 23 de Octubre de 2015. URL: http://mininet.org/download

[32] Mininet (2015) Mininet VM Setup Notes, Mininet. Fecha de consulta: 23 de Octubre de 2015. URL: http://mininet.org/vm-setup-notes

[33] GitHub (2015) VirtualBox specific Instructions, GitHub. Fecha de consulta: 23 de Octubre de 2015. URL: https://github.com/mininet/openflow-tutorial/wiki/VirtualBox-specific-Instructions

[34] Brian Linkletter (17/09/2013) Set up the Mininet network simulator, Open Source Routing and Network Simulation blog. Fecha de consulta: 23 de Octubre de 2015. URL: http://www.brianlinkletter.com/set-up-mininet

[35] XQuartz (2015) A version of the X.Org X Window System that runs on OS X, XQuartz. Fecha de consulta: 25 de Octubre de 2015. URL: http://xquartz.macosforge.org/landing

[36] Lantz (02/04/2015) FAQ, GitHub. Fecha de consulta: 26 de Octubre de 2015. URL: https://github.com/mininet/mininet/wiki/FAQ#examples

[37] Lantz (25/06/2015) mininet/examples, GitHub. Fecha de consulta: 26 de Octubre de 2015. URL: https://github.com/mininet/mininet/tree/master/examples

[38] OpenFlow Tutorial, OpenFlow. Fecha de consulta: 27 de Octubre de 2015. URL: http://archive.openflow.org/wk/index.php/OpenFlow_Tutorial#Set_up_Virtual_Machine

[39] McCauley Murphy (05/03/2015) POX Wiki, OpenFlow. Fecha de consulta: 30 de Octubre de 2015. URL: https://openflow.stanford.edu/display/ONL/POX+Wiki#POXWiki-misc.mac_blocker

[40] Mininet (2015) Using a Remote Controller, Mininet. Fecha de consulta: 30 de Octubre de 2015. URL: http://mininet.org/walkthrough/#using-a-remote-controller

[41] OpenFlow Tutorial with POX - Part 1, Python for Network Engineers. Fecha de consulta: 04 de Noviembre de 2015. URL: http://blog.pythonicneteng.com/2013/02/openflow-tutorial-with-pox.html

[42] Raaj Akshar (03/06/2012) Understanding '*', '*args', '**' and '**kwargs'. Fecha de consulta: 28 de Noviembre de 2015. URL: http://agiliq.com/blog/2012/06/understanding-args-and-kwargs

Anexos

Anexo 1

Parece que los enlaces eliminan los paquetes en ambas direcciones (envío y recepción), es decir:

- Un 4% de las peticiones que van de h1 a s1
- Un 4% de las peticiones que van de s1 a h2
- Un 4% de las respuestas que van de h2 a s1
- Un 4% de las respuestas que van de s1 a h2

En la siguiente tabla se puede observar como estos resultados teóricos concuerdan aproximadamente con los resultados reales obtenidos en los pings. A la hora de contabilizar paquetes sólo se tiene en cuenta la parte entera. Es posible que estos decimales sean lo que haga que no concuerden exactamente los resultados.
Esto demuestra que no se elimina el 4% de los paquetes para cada IP de origen, tal y como se pretendía en este caso de uso.

Salen de h1	Llegan a s1 (el enlace ha eliminado el 4%)	Llegan a h2 y contesta (el enlace ha eliminado el 4%)	Llegan a s1 (el enlace ha eliminado el 4%)	Llegan a h1 (el enlace ha eliminado el 4%)	Resultado TEST	Diferencia
25	24	23,04 ≈ 23	22,08 ≈ 22	21,12 ≈ 21	20	1
26	24,96 ≈ 24	23,04 ≈ 23	22,08 ≈ 22	21,12 ≈ 21	21	0
27	25,92 ≈ 25	24	23,04 ≈ 23	22,08 ≈ 22	26	4
50	48	46,08 ≈ 46	44,16 ≈ 44	42,24 ≈ 42	41	1
51	48,96 ≈ 48	46,08 ≈ 46	44,16 ≈ 44	42,24 ≈ 42	46	4
52	49,92 ≈ 49	47,04 ≈ 47	45,12 ≈ 45	43,2 ≈43	42	1
75	72	69,12 ≈ 69	66,24 ≈ 66	63,36 ≈ 63	64	1
76	72,96 ≈ 72	69,12 ≈ 69	66,24 ≈ 66	63,36 ≈ 63	70	7
77	73,92 ≈ 73	70,08 ≈ 70	67,2 ≈ 67	64,32 ≈ 64	68	4
100	96	92,16 ≈ 92	88,32 ≈ 88	84,48 ≈ 84	82	2
250	240	230,4 ≈230	220,8 ≈ 220	211,2 ≈ 211	211	0
500	480	460,8 ≈ 460	441,6 ≈ 441	423,36 ≈ 423	425	2
750	720	691,2 ≈ 691	663,36 ≈ 663	636,48 ≈ 636	645	9
1000	960	921,6 ≈ 921	884,16 ≈ 884	848,64 ≈848	855	7
1250	1200	1152	1105,92 ≈ 1105	1060,8 ≈ 1060	1065	5
1500	1440	1382,4 ≈ 1382	1326,72 ≈ 1326	1272,96 ≈ 1272	1281	9

Tabla Anexo 1 Comparativa resultados teóricos y resultados reales (sol.1)

Anexo 2

Código del componente "of_tutorialLoos.py" que se ha implementado para el controlador POX con el fin de solucionar el caso de uso 1, solución 2 (para cada IP de origen, eliminar 1 de cada 25 paquetes que llegan al controlador).
El código en negrita es el código que se ha añadido al script original.

```
# Copyright 2012 James McCauley
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at:
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
# See the License for the specific language governing permissions and
# limitations under the License.

"""
This component is for use with the OpenFlow tutorial.

It acts as a simple hub, but can be modified to act like an L2
learning switch.

It's roughly similar to the one Brandon Heller did for NOX.
"""

from pox.core import core
import pox.openflow.libopenflow_01 as of

log = core.getLogger()


class Tutorial (object):
  """
  A Tutorial object is created for each switch that connects.
  A Connection object for that switch is passed to the __init__ function.
  """
  def __init__ (self, connection):
    # Keep track of the connection to the switch so that we can
    # send it messages!
    self.connection = connection

    # This binds our PacketIn event listener
    connection.addListeners(self)



    # Use this table to keep track of which ethernet address is on
    # which switch port (keys are MACs, values are ports).
    self.mac_to_port = {}
```

```
# Tabla para guardar las IPs de origen y sus contadores de paquetes (las claves son IPs y los valores son los contadores)
    self.IP_Paquetes = {}

  def resend_packet (self, packet_in, out_port):
    """
    Instructs the switch to resend a packet that it had sent to us.
    "packet_in" is the ofp_packet_in object the switch had sent to the controller due to a table-miss.
    """
    msg = of.ofp_packet_out()
    msg.data = packet_in

    # Add an action to send to the specified port
    action = of.ofp_action_output(port = out_port)
    msg.actions.append(action)

    # Send message to switch
    self.connection.send(msg)

# Código adicional para filtrar los paquetes y eliminar el 4% de los paquetes de cada IP de origen
def eliminaPaquete (self, packet):
    """
    Comprueba si la IP de origen del paquete ya ha llegado a 25 envíos o no. Y si ha llegado devuelve True.
    """
# Se comprueba si el paquete es de tipo IP
    if packet.type == packet.IP_TYPE:
# Si es de tipo IP se guarda su IP de origen
      ip = packet.next.srcip
# Se comprueba si la IP está en la tabla que guarda las IP y los paquetes enviados
      if ip in self.IP_Paquetes:
# Si la IP está en la tabla se obtiene su contador de paquetes enviados y se incrementa
        cont = self.IP_Paquetes[ip]
        cont+=1
# Se comprueba el contador de paquetes enviados a llegado a 25
        if cont == 25:
# Si el contador de paquetes enviados ha llegado a 25 se pone a cero y se devuelve True (hay que eliminar el paquete)
          cont = 0
          self.IP_Paquetes[ip] = cont
          return True
        else:
# Si el contador de paquetes enviados no ha llegado a 25 se guarda el contador y se devuelve False (hay que enviar el paquete)
          self.IP_Paquetes[ip] = cont
          return False
      else:
# Si la IP de origen no está en la tabla de IPs y contadores se inicializa el contador a 1 y se devuelve False (hay que enviar el paquete)
        cont = 1
```

```
        self.IP_Paquetes[ip] = cont
        return False
    else:
# Si el paquete no es de tipo IP se devuelve False (hay que enviar el paquete)
      return False


def act_like_hub (self, packet, packet_in):
    """
    Implement hub-like behavior -- send all packets to all ports besides
    the input port.
    """

    # We want to output to all ports -- we do that using the special
    # OFPP_ALL port as the output port.  (We could have also used
    # OFPP_FLOOD.)

# Si la IP no ha enviado aun 25 paquetes, se envía el paquete
    if self.eliminaPaquete(packet) == False:
      self.resend_packet(packet_in, of.OFPP_ALL)

    # Note that if we didn't get a valid buffer_id, a slightly better
    # implementation would check that we got the full data before
    # sending it (len(packet_in.data) should be == packet_in.total_len)).

def act_like_switch (self, packet, packet_in):
    """
    Implement switch-like behavior.
    """
    """ # DELETE THIS LINE TO START WORKING ON THIS (AND THE ONE
BELOW!) #

    # Here's some psuedocode to start you off implementing a learning
    # switch.  You'll need to rewrite it as real Python code.

    # Learn the port for the source MAC
    self.mac_to_port ... <add or update entry>

    if the port associated with the destination MAC of the packet is known:
      # Send packet out the associated port
      self.resend_packet(packet_in, ...)

      # Once you have the above working, try pushing a flow entry
      # instead of resending the packet (comment out the above and
      # uncomment and complete the below.)

      log.debug("Installing flow...")
      # Maybe the log statement should have source/destination/port?

      #msg = of.ofp_flow_mod()
      #
      ## Set fields to match received packet
      #msg.match = of.ofp_match.from_packet(packet)
      #
      #< Set other fields of flow_mod (timeouts? buffer_id?) >
```

```
      #
      #< Add an output action, and send -- similar to resend_packet() >
    else:
      # Flood the packet out everything but the input port
      # This part looks familiar, right?
      self.resend_packet(packet_in, of.OFPP_ALL)

    """ # DELETE THIS LINE TO START WORKING ON THIS #

  def _handle_PacketIn (self, event):
    """
    Handles packet in messages from the switch.
    """
    packet = event.parsed # This is the parsed packet data.
    if not packet.parsed:
      log.warning("Ignoring incomplete packet")
      return

    packet_in = event.ofp # The actual ofp_packet_in message.

    # Comment out the following line and uncomment the one after
    # when starting the exercise.
    self.act_like_hub(packet, packet_in)
    #self.act_like_switch(packet, packet_in)

def launch ():
  """

  Starts the component
  """
  def start_switch (event):
    log.debug("Controlling %s" % (event.connection,))
    Tutorial(event.connection)
  core.openflow.addListenerByName("ConnectionUp", start_switch)
```

Anexo 3

En la siguiente tabla se muestra de forma detallada la evolución de los contadores y los bloqueos que se producen, por parte del componente implementado para el controlador POX (caso de uso 1, solución 2), mientras se van realizando las diferentes peticiones de ping desde el host 1 al host 2.

Ping		Contador Host 1 (petición)	Contador Host 2 (respuesta)		Paquetes perdidos	Perdidos acumulados
1		1	1			
2		2	2			
…		…	…			
23		23	23			
24		24	24			
25	**Bloqueo, no se envía la petición**	**25**		**No llega la petición, no se envía respuesta**	**1 perdido**	**1 perdido**
26		1	25	Bloqueo, no se envía la respuesta	1 perdido	2 perdidos
27		2	1			
28		3	2			
…		…	…			
48		23	22			
49		24	23			
50	**Bloqueo, no se envía la petición**	**25**		**No llega la petición, no se envía respuesta**	**1 perdido**	**3 perdidos**
51		1	24			
52		2	25	Bloqueo, no se envía la respuesta	1 perdido	4 perdidos
53		3	1			
54		4	2			
…		…	…			
73		23	21			
74		24	22			
75	**Bloqueo, no se envía la petición**	**25**		**No llega la petición, no se envía respuesta**	**1 perdido**	**5 perdidos**
76		1	23			
77		2	24			
78		3	25	Bloqueo, no se envía la respuesta	1 perdido	6 perdidos
79		4	1			
80		5	2			
98		23	20			

99		24	21			
100	**Bloqueo, no se envía la petición**	**25**		**No llega la petición, no se envía respuesta**	**1 perdido**	**7 perdidos**
…						

Tabla Anexo 3 Evolución de los contadores y bloqueos durante los pings (sol.2)

Anexo 4

Código del script "topotc.py" que se ha implementado para crear la topología con el fin de solucionar el caso de uso 2, solución 1 (para una determinada IP indicada por parámetro, eliminar el 4% del su tráfico).
El código en negrita es el código que se ha añadido al script original.

```
#!/usr/bin/python

"""
Simple example of setting network parameters
NOTE: link param limit loss.
"""
from mininet.topo import Topo
from mininet.net import Mininet
from mininet.link import TCLink
from mininet.log import setLogLevel
from mininet.cli import CLI
import sys

#Se recoge la IP que se ha pasado como parámetro
IPbloqueada = sys.argv[1]

class SingleSwitchTopo(Topo):
    "Single switch connected to n hosts."
    def __init__(self, n=2, **opts):
        Topo.__init__(self, **opts)
        switch = self.addSwitch('s1')
        for h in range(n):
            host = self.addHost('h%s' % (h + 1))
            self.addLink(host, switch)

def perfTest():
    "Creación de la red con una perdida del 4% en la interface del host con la IP
indicada"
    topo = SingleSwitchTopo( n=2 )
    net = Mininet( topo=topo, link=TCLink, autoStaticArp=True )
    net.start()

# Código adicional para filtrar los paquetes y eliminar el 4% de los paquetes de
cada IP de origen
    hosts = net.hosts
#Se busca el host con la IP indicada
    for h in hosts:
        if h.IP() == IPbloqueada:
          IPhost = h.IP()
          nombreHost = h.name
          print "Bloqueo del 4% al host " + nombreHost + " que tiene la IP " + IPhost
#Se fuerza una perdida del 4% en la interface del host que tiene la IP indicada
por parámetro
          h.cmd('tc qdisc add dev ' + nombreHost + '-eth0 root netem loss 4%')

    CLI(net)
    net.stop()
```

```
if __name__ == '__main__':
    setLogLevel('info')
    perfTest()
```

Anexo 5

Parece que el enlace elimina los paquetes en una dirección (envío o recepción).

En la siguiente tabla se puede observar como estos resultados teóricos concuerdan aproximadamente con los resultados reales obtenidos en los pings. A la hora de contabilizar paquetes sólo se tiene en cuenta la parte entera.
Esto demuestra que no se elimina exactamente el 4% y que el comportamiento de los paquetes no es regular.

Salen de h1	El enlace ha eliminado el 4%	Resultado TEST	Diferencia
25	24	24	0
26	24,96 ≈ 24	23	1
27	25,92 ≈ 25	27	2
50	48	46	2
51	48,96 ≈ 48	50	2
52	49,92 ≈ 49	49	0
75	72	74	2
76	72,96 ≈ 72	75	3
77	73,92 ≈ 73	74	1
100	96	97	1
250	240	235	5
500	480	481	1
750	720	725	5
1000	960	957	3
1250	1200	1206	6
1500	1440	1440	0

Tabla Anexo 5 Comparativa resultados teóricos y resultados reales (sol.1)

Anexo 6

Código del componente "of_tutorialLoos2.py" que se ha implementado para el controlador POX con el fin de solucionar el caso de uso 2, solución 2 (para una determinada IP de origen indicada por parámetro, eliminar 1 de cada 25 paquetes que llegan al controlador). [42]
El código en negrita es el código que se ha añadido al script original.

```
# Copyright 2012 James McCauley
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at:
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
# See the License for the specific language governing permissions and
# limitations under the License.

"""
This component is for use with the OpenFlow tutorial.

It acts as a simple hub, but can be modified to act like an L2
learning switch.

It's roughly similar to the one Brandon Heller did for NOX.
"""

from pox.core import core
import pox.openflow.libopenflow_01 as of

log = core.getLogger()

class Tutorial (object):
  """
  A Tutorial object is created for each switch that connects.
  A Connection object for that switch is passed to the __init__ function.
  """
  # Con el parámetro IP se indica que se espera este parámetro
  def __init__ (self, connection, IP):
    # Keep track of the connection to the switch so that we can
    # send it messages!
    self.connection = connection

    # This binds our PacketIn event listener
    connection.addListeners(self)

    # Use this table to keep track of which ethernet address is on
    # which switch port (keys are MACs, values are ports).
    self.mac_to_port = {}
```

```
    # Variable para guardar el numero de paquetes que lleva enviados la IP
indicada
    self.IP_Paquetes = 0
    # Variable para guardar el numero de paquetes que se llevan enviados (a nivel
informativo)
    self.paquetes = 0
    # Variabla para guardar la IP que se ha indicado como parámetro
    self.IPbloqueada = IP

  def resend_packet (self, packet_in, out_port):
    """
    Instructs the switch to resend a packet that it had sent to us.
    "packet_in" is the ofp_packet_in object the switch had sent to the
    controller due to a table-miss.
    """
    msg = of.ofp_packet_out()
    msg.data = packet_in

    # Add an action to send to the specified port
    action = of.ofp_action_output(port = out_port)
    msg.actions.append(action)

    # Send message to switch
    self.connection.send(msg)


# Código adicional para eliminar un 4% de los paquetes, uno de cada 25
  def eliminaPaquete (self, packet):
    """
    Comprueba si la IP indicada ya ha llegado a 25 paquetes enviados o no
    """
    if packet.type == packet.IP_TYPE:
      ip = packet.next.srcip
      self.paquetes+=1
      print("Packet tipo IP numero: ", self.paquetes)
      print("Packet con IP: ", ip)
# Se comprueba si la IP de origen del paquete es igual a la IP indicada
      if str(ip) == self.IPbloqueada:
# Si el paquete lo envía la IP indicada, se incrementa su contador de paquetes
enviados
        self.IP_Paquetes+=1
        print("IP indicada y su contador es: ", self.IP_Paquetes)
# Se comprueba si la IP indicada ha llegado ya a los 25 paquetes enviados
        if self.IP_Paquetes == 25:
# Si ha llegado a 25 paquetes enviados se pone el contador a cero y se devuelve
True (hay que eliminar el paquete)
          self.IP_Paquetes = 0
          print("IP bloqueada: ", ip)
          return True
        else:
# Si no ha llegado a 25 paquetes enviados se devuelve False (hay que enviar el
paquete)
          return False
      else:
# Si el paquetes no lo envía la IP indicada se devuelve False (hay que enviar el
```

```
paquete)
        print("IP no indicada")
        return False
    else:
# Si el paquete no es de tipo IP se devuelve False (hay que enviar el paquete)
      return False

def act_like_hub (self, packet, packet_in):
    """
    Implement hub-like behavior -- send all packets to all ports besides
    the input port.
    """

    # We want to output to all ports -- we do that using the special
    # OFPP_ALL port as the output port.  (We could have also used
    # OFPP_FLOOD.)

# Si la IP no ha enviado aun 25 paquetes, se envía el paquete
    if self.eliminaPaquete(packet) == False:
      self.resend_packet(packet_in, of.OFPP_ALL)

    # Note that if we didn't get a valid buffer_id, a slightly better
    # implementation would check that we got the full data before
    # sending it (len(packet_in.data) should be == packet_in.total_len)).

def act_like_switch (self, packet, packet_in):
    """
    Implement switch-like behavior.
    """

    """ # DELETE THIS LINE TO START WORKING ON THIS (AND THE ONE
BELOW!) #

    # Here's some psuedocode to start you off implementing a learning
    # switch.  You'll need to rewrite it as real Python code.

    # Learn the port for the source MAC
    self.mac_to_port ... <add or update entry>

    if the port associated with the destination MAC of the packet is known:
      # Send packet out the associated port
      self.resend_packet(packet_in, ...)

      # Once you have the above working, try pushing a flow entry
      # instead of resending the packet (comment out the above and
      # uncomment and complete the below.)

      log.debug("Installing flow...")
      # Maybe the log statement should have source/destination/port?

      #msg = of.ofp_flow_mod()
      #
      ## Set fields to match received packet
      #msg.match = of.ofp_match.from_packet(packet)
      #
```

```
      #< Set other fields of flow_mod (timeouts? buffer_id?) >
      #
      #< Add an output action, and send -- similar to resend_packet() >

    else:
      # Flood the packet out everything but the input port
      # This part looks familiar, right?
      self.resend_packet(packet_in, of.OFPP_ALL)

    """ # DELETE THIS LINE TO START WORKING ON THIS #

def _handle_PacketIn (self, event):
    """
    Handles packet in messages from the switch.
    """

    packet = event.parsed # This is the parsed packet data.
    if not packet.parsed:
      log.warning("Ignoring incomplete packet")
      return

    packet_in = event.ofp # The actual ofp_packet_in message.

    # Comment out the following line and uncomment the one after
    # when starting the exercise.
    self.act_like_hub(packet, packet_in)
    #self.act_like_switch(packet, packet_in)

# Con le parámetro IP se indica que se espera este parámetro
def launch (IP):
  """
  Starts the component
  """
  def start_switch (event):
    log.debug("Controlling %s" % (event.connection,))
    Tutorial(event.connection, IP)
  core.openflow.addListenerByName("ConnectionUp", start_switch)
```