Gestión de dispositivos IoT con acceso a la red mediante 802.11ax y SDN en el edge para orquestar los flujos de datos

Sergio Baza Alonso
Máster en Ingeniería de Telecomunicaciones
Área Telemática

Jose Lopez Vicario
Xavi Vilajosana Guillen

03/01/2021

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Gestión de dispositivos IoT con acceso a la red mediante 802.11ax y SDN en el edge para orquestar los flujos de datos*
Nombre del autor:	*Sergio Baza Alonso*
Nombre del consultor/a:	*Jose Lopez Vicario*
Nombre del PRA:	*Xavi Vilajosana Guillen*
Fecha de entrega (mm/aaaa):	01/2021
Titulación::	*Máster Ingeniería de Telecomunicaciones*
Área del Trabajo Final:	*Telemática*
Idioma del trabajo:	*Castellano*
Palabras clave	*SDN, NFV, WiFi*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

Con la llegada de la Industria 4.0 y con el crecimiento del número de dispositivos IoT en este entorno se hace necesario controlar la forma de acceso de estos dispositivos a la red. Debido a la gran cantidad de dispositivos IoT existentes, se hace más conveniente conectarlos a la red mediante una forma de acceso inalámbrica.

Existen dos maneras principales de dar acceso a la red de manera inalámbrica, mediante redes WLAN privadas (WiFi) y mediante redes móviles públicas (LTE, 5G). Ambos dos capaces de proporcionar conectividad de gran velocidad a cualquier tipo de dispositivo.

La Industria 4.0 tiene varios ámbitos de desarrollo, pero el enfoque de este trabajo se centra en el sector privado en el entorno industrial. Es debido a la estricta normativa de seguridad que imponen las empresas privadas que se ha escogido la vía de acceso mediante WiFi, una red privada desde la que se puedan controlar los accesos de cada dispositivo.

Por último, la suma de las redes definidas por software (SDN) y la virtualización de funciones de red (NFV) están llamadas a sustituir, simplificar y optimizar las redes tal y como las conocemos hoy en día, por lo que su extensión a entornos industriales es inevitable.

El proyecto por tanto consiste en la aplicación de SDN a un entorno industrial con dispositivos IoT y la orquestación de los flujos de datos de estos últimos estando conectados mediante WiFi 6.

Abstract (in English, 250 words or less):

With the arrival of Industry 4.0 and the growth in the number of IoT devices in this environment, it becomes necessary to control the way these devices access the network. Due to the large number of existing IoT devices, it becomes more convenient to connect them to the network through a form of wireless access.

There are two main ways to provide wireless network access, via private WLAN (WiFi) and via public mobile networks (LTE, 5G). Both are capable of providing high-speed connectivity to any type of device.

Industry 4.0 has several areas of development, but the focus of this thesis is on the private sector in the industrial environment. It is due to the strict security regulations imposed by private companies that the access route via WiFi, a private network from which access to each device can be controlled, has been chosen.

Finally, the sum of software-defined networks (SDN) and network function virtualisation (NFV) are called upon to replace, simplify and optimise networks as we know them today, making their extension to industrial environments inevitable.

The project therefore consists of applying SDN to an industrial environment with IoT devices and orchestrating the data flows of the latter while connected via WiFi 6.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

Hoy en día cada vez es más frecuente encontrarse con que todos los dispositivos necesitan o son capaces de tener una conexión a internet para explotar totalmente todas sus posibilidades. Este tipo de dispositivos (IoT) están en todo tipo de entornos, incluyendo los industriales y es necesario dotarlos de conexión a internet. Debido a la gran cantidad de estos dispositivos se hace inconveniente conectarlos a la red mediante cable y es preferible optar por las tecnologías inalámbricas.
Paralelamente a esto, las redes están cambiando por primera vez desde hace mucho tiempo gracias al paradigma de las redes definidas por software (SDN). A través de SDN se consigue que las redes no sean más que meras infraestructuras gobernadas por un equipo controlador, que construye una red virtual sobre la red física. Es este controlador el que se encarga de indicar a todos los elementos de la red como deben actuar ante la llegada de cualquier flujo de datos.
Adicionalmente a todo lo anterior, los estándares de acceso inalámbrico al medio están evolucionando también debido a las ventajas que presentan frente a los medios de acceso por cable. Los estándares de acceso inalámbricos proporcionan una conexión más cómoda y barata de realizar por el mero hecho de no necesitar un cable para tener acceso a la red. La desventaja que tienen estos estándares es el ancho de banda que son capaces de proporcionar a cada cliente y la necesidad de tener controlados otros aspectos (interferencias, clientes en movimiento, etc...) Actualmente hay dos maneras principales de acceder a la red de forma inalámbrica, mediante WiFi (el último estándar es 802.11ax/WiFi 6) o mediante un operador móvil (LTE/5G). Debido a que lo considero más adecuado para entornos no públicos el modo de acceso en el que voy a centrarme es WiFi, a pesar de que ambos dos serían válidos y proporcionarían un rendimiento similar.
Adicionalmente a esto y por lo general las redes SDN han estado siempre más orientadas a entornos cableados, pero las ventajas que proporcionan este tipo de redes pueden aplicarse a cualquier tipo de red. Actualmente las redes que tienen mayor protagonismo son las cableadas y si se aplica SDN en las empresas se aplica principalmente en la parte WAN empresarial (SD-WAN), pero las redes WiFi cada vez tienen mayor protagonismo. Viendo las capacidades de red que son capaces de ofertar los nuevos estándares inalámbricos no es descabellado pensar que las oficinas del futuro no tendrán cables, sólo tendran los necesarios para interconectar los switches de las distintas capas (acceso, distribución y core). Es por ello que se ha considerado interesante profundizar en la aplicación de SDN sobre este tipo de redes, muy probablemente en el futuro su uso esté más extendido.
Este trabajo pretende definir como orquestar los flujos de datos que generan los dispositivos IoT dentro de una red SDN, partiendo de que la forma de acceso de estos dispositivos a la red sea inalámbrica (802.11ax).

1.2 Objetivos del Trabajo

De cara al trabajo los objetivos y subobjetivos que los componen son:

- Aplicación de SDN en un entorno empresarial de caracter industrial con gran cantidad de dispositivos IoT conectados de manera inalámbrica.
    - Realizar un estudio de la situación tecnológica de redes SDN sobre una infraestructura inalámbrica (SD-WLAN).
    - Aplicación de herramientas de virtualización de funciones de red (NFV) y orquestación de servicios para sustituir a los servicios que oferta una red inalámbrica tradicional.

- Realización de un estudio del nuevo estándar inalámbrico 802.11ax
    - Comprobar las nuevas funcionalidades del estándar para ver si ofrece ventajas a los dispositivos IoT frente a los estándares anteriores.
    - Realizar un estudio del tipo de tramas que se generan en un entorno inalámbrico para elegir la mejor manera de integrarlos en un entorno SDN.

- Diseño de una arquitectura de red que permita orquestar los flujos de datos provenientes de una red de acceso inalámbrica mediante SDN.

- Evaluación de la solución mediante una maqueta y análisis de los resultados obtenidos.

1.3 Enfoque y método seguido

El enfoque del trabajo que se va a utilizar consiste en dividir el objetivo final en cinco fases. El objetivo final del proyecto es diseñar una arquitectura de red basada en SDN con funciones de red virtuales que permita gestionar dispositivos conectados inalámbricamente. Esta arquitectura se compone de tres aspectos:

- Acceso inalámbrico mediante WiFi.
- Componentes de la red:
    - Controlador SDN
    - Switches capaces de trabajar con OpenFlow
    - APs capaces de trabajar con OpenFlow y/o que se integren con la función virtual de red que se use como controlador inalámbrico.
- Elementos de virtualización:
    - Funciones de red virtualizadas
    - Hypervisor para despliegue de infraestructura virtual

Es por tanto que el trabajo primero se centra en cada uno de los aspectos y en describir cuál es la situación tecnológica de cada uno de ellos.

En la primera fase se pretenden abordar los aspectos relacionados con SDN, arquitectura y funcionamiento incluyendo el protocolo OpenFlow. Dentro de

este apartado se pretende también realizar una comparación entre diferentes controladores SDN para posteriormente decidir cuál es más adecuado.

En la segunda fase se abordan los aspectos relacionados con la virtualización. Por un lado, la virtualización de la infraestructura, ya que la arquitectura final requiere el despliegue de máquinas virtuales. Por otro lado, la virtualización de funciones de red ya que es necesario gestionar el entorno inalámbrico a través de funciones de red virtuales.

En la tercera fase se estudia todo lo relacionado con la vía de acceso por WiFi. Es necesario recabar información acerca del funcionamiento del estándar. Hay que tener conocimiento acerca de arquitecturas de redes inalámbricas tradicionales y también acerca de las tramas que se generan en estos entornos. Conociendo la arquitectura y las tramas se puede decidir cómo adaptar estas arquitecturas a funciones de red virtuales. Por último, se busca conocer el estado actual de la gestión de redes inalámbricas a través de SDN.

La cuarta fase comprende el diseño de la arquitectura en conjunto. Una vez estudiados todos los elementos por separado ya se estará en disposición de elegir los componentes que componen cada uno de los aspectos de la arquitectura. Se elegirán dichos componentes y se establecerá como serán las comunicaciones entre ellos.

La quinta y última fase consiste en llevar a la práctica la arquitectura propuesta mediante una combinación de recursos virtuales y físicos. Dentro de esta fase se pretende comprobar la validez de la solución propuesta.

Para realizar pruebas sobre el entorno se proponen tres casos de uso.

- Primer caso de uso: Conseguir comunicación entre dos dispositivos IoT conectados por WiFi. Se realizará el aprovisionamiento de uno o varios APs donde conectar dos dispositivos IoT. Se comprobará que hay comunicación entre los dispositivos.

- Segundo caso de uso: Conseguir comunicación entre un dispositivo IoT conectado por WiFi y un dispositivo conectado mediante cable de red. Se realizará el aprovisionamiento de un punto de acceso para permitir la conexión de un dispositivo IoT y conectará un equipo por cable a uno de los switches. Se instalarán flujos OpenFlow en los switches y se comprobará que hay comunicación entre los dispositivos.

- Tercer caso de uso: Bloquear tráfico entre dos dispositivos, uno de ellos IoT y otro de ellos conectado a la red por cable. Se realizará el aprovisionamiento de un punto de acceso para permitir la conexión de un dispositivo IoT y conectará un equipo por cable a uno de los switches. Se comprobará que hay comunicación entre ambos y se instalará un flujo en los switches OpenFlow para impedir esta comunicación.

1.4 Planificación del Trabajo

Para la realización de este trabajo son necesarios una serie de recursos que posibilitan la puesta en marcha de la arquitectura de red sobre la que probar los diferentes casos de uso propuestos.
Los recursos son los siguientes:

- Software VirtualBox para la creación de máquinas virtuales.
- Software GNS3 sobre el que montar la maqueta de arquitectura de red.
- Sistema operativo Lubuntu que se usará de base para instalar los siguientes elementos:
    - Controlador Ryu
    - Switch OpenFlow: Open vSwitch
    - Servidor Free Radius
- Imagen RouterOS de Mikrotik para la virtualización del controlador WiFi y los APs.
- Router físico Mikrotik para ser utilizado como AP.
- Raspberry Pi para ser utilizada como dispositivo IoT.
- Teléfono móvil para ser utilizado como dispositivo IoT.
- Equipo sobre el que realizar la virtualización, HP EliteBook 840 G5, procesador i7, 8Gb RAM.
- Cable UTP categoría 5E.

Dentro de las tareas planificadas más relevantes se muestra a continuación la evaluación de ellas según lo planificado.

- Fase 1: Estudio sobre redes SDN

En esta fase se han llevado a cabo labores de estudio e investigación de esta tecnología y los diferentes protocolos que se utilizan en la misma. Hay una gran cantidad de información disponible por lo que es necesario analizarla y recoger sólo los aspectos que puedan resultar más relevantes. Para esta fase se han utilizado 15 horas.

- Fase 2: Estudio sobre virtualización de funciones de red

Durante esta fase se han estudiado varias herramientas de virtualización de funciones de red y metodologías para poder virtualizar equipos como funciones de red. También se ha hecho énfasis en la virtualización de funciones de red dentro de redes inalámbricas. Para esta fase se han utiliado 10 horas.

- Fase 3: Estudio sobre acceso inalámbrico y cómo se aplica a SDN

En la tercera fase se aborda el estudio de todo lo relacionado con el acceso a la red mediante tecnología inalámbrica. Se pone especial interés en el nuevo estándar inalámbrico 802.11ax y los beneficios que aporta sobre dispositivos IoT. Otro aspecto que queda cubierto en esta fase es el estudio de la aplicación actual de SDN a redes inalámbricas. Se usan en total 16 horas para esta fase.

- Fase 4: Diseño de la arquitectura de red completa

En esta fase se aborda el diseño de la red. Una vez completadas las fases anteriores ya se está en disposición de elegir las herramientas que permitan realizar el diseño de la red. Para esta fase se han utilizado en total 12 horas de las 15 horas previstas.

- Fase 5: Pruebas sobre escenario virtual

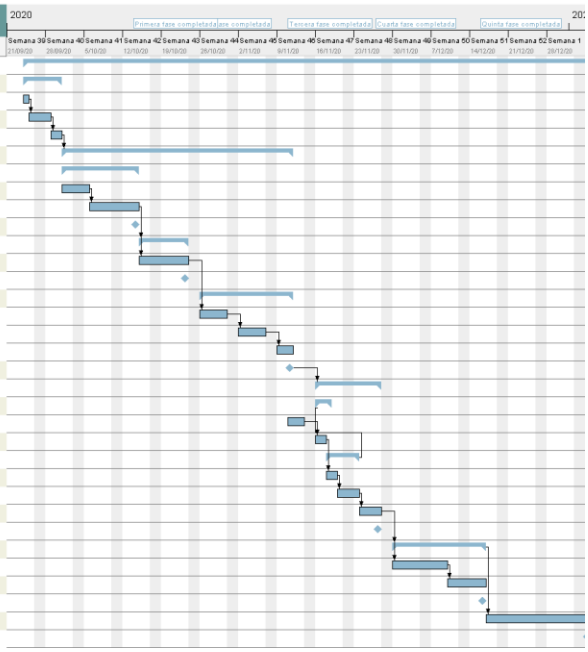
Durante esta fase se aborda el despliegue de la arquitectura de red propuesta sobre una maqueta virtual de GNS3. Una vez creada la maqueta virtual es necesario realizar la configuración de la misma para garantizar la comunicación entre los distintos dispositivos que la componen. Con la maqueta montada se desarrollan las aplicaciones necesarias para el funcionamiento de la red inalámbrica y se prueban los tres casos de uso propuestos. Para esta fase la estimación de tiempo dedicado son 25 horas.

- Fase 6: Documentación realizada y fin de proyecto

La última fase del proyecto consiste en la redacción final del documento del trabajo. Para esta fase se estima que serán necesarias unas 20 horas.

El diagrama de Gantt de la planificación del proyecto es el siguiente:

Nombre	Fecha de inicio	Fecha de fin
TFM	24/09/20	4/01/21
0. Definiciones iniciales	24/09/20	30/09/20
Definición del contexto	24/09/20	24/09/20
Definición de objetivos	25/09/20	28/09/20
Definición de alcance	29/09/20	30/09/20
1. Definición de la situación tecnológica	1/10/20	11/11/20
Redes definidas por Software	1/10/20	14/10/20
Protocolo OpenFlow	1/10/20	5/10/20
Controlador SDN	6/10/20	14/10/20
Primera fase completada	14/10/20	14/10/20
Virtualización de funciones de red	15/10/20	23/10/20
Herramientas de orquestación NFV	15/10/20	23/10/20
Segunda fase completada	23/10/20	23/10/20
Acceso inalámbrico IEEE 802.11	26/10/20	11/11/20
Estudio de las tramas usadas por el estándar	26/10/20	30/10/20
Estudio de arquitecturas inalámbricas convencionales	2/11/20	6/11/20
Aplicación de SDN a redes WiFi	9/11/20	11/11/20
Tercera fase completada	11/11/20	11/11/20
2. Diseño de la arquitectura	16/11/20	27/11/20
Diseño NFV	16/11/20	18/11/20
Elección herramienta orquestación/NFV	11/11/20	13/11/20
Elección de la herramienta de VIM	16/11/20	17/11/20
Diseño SDN	18/11/20	23/11/20
Elección del controlador SDN	18/11/20	19/11/20
Elección de los switches OpenFlow y APs WiFi	20/11/20	23/11/20
Integración de los elementos	24/11/20	27/11/20
Cuarta fase completada	27/11/20	27/11/20
3. Montaje escenario de pruebas	30/11/20	16/12/20
Despliegue máquinas virtuales	30/11/20	9/12/20
Pruebas sobre escenario de pruebas	10/12/20	16/12/20
Quinta fase completada	16/12/20	16/12/20
4. Redacción de la documentación del proyecto	17/12/20	4/01/21
Fin del proyecto	4/01/21	4/01/21



En cuanto a los hitos se ha considerado un hito por cada una de las fases propuestas completadas.

1.5 Breve sumario de productos obtenidos

Debido al proyecto y la metodología los productos obtenidos de este trabajo estarán basados en los resultados obtenidos al simular los tres casos de uso propuestos sobre la maqueta de red.

1.6 Dificultades encontradas

En la ejecución de este proyecto se han encontrado estas dificultades:

- Recursos físicos de memoria bajos en el equipo que aloja la maqueta. Inicialmente estaba planificado el uso del controlador ONOS y 7 switches OpenFlow. Debido a que ha sido más relevante asignar recursos a la máquina de virtualización de funciones de red ha sido necesario utilizar menos switches y otro controlador SDN.

- Inicialmente se planteaba el uso de dos puntos de acceso diferenciados, uno de ellos físico y otro virtual. No se ha encontrado manera de simular una tarjeta inalámbrica dentro de los equipos Mikrotik virtuales, por lo que se ha tenido que utilizar un solo punto de acceso físico.

1.7 Breve descripción de los otros capítulos de la memoria

El desarrollo de este trabajo se lleva a cabo en los siguientes capítulos:

Capítulo 2 – Situación tecnológica

En este capítulo se describe la situación tecnológica actual de las tecnologías relevantes en este trabajo (SDN, OpenFlow, NFV, WiFi). A parte de estudiar cada tecnología por separado se estudia el estado actual de aplicación de SDN a redes inalámbricas y cómo son las arquitecturas de redes inalámbricas tradicionales. Por último se hace un repaso a varias aportaciones de otras personas en trabajos con finalidad similar.

Capítulo 3 – Diseño de la solución

En este capítulo se eligen los diferentes elementos que van a componer la arquitectura de red una vez que se han estudiado todos los elementos por separado y se está en disposición de elegirlos de manera justificada.

Capítulo 4 – Escenario de pruebas

En este capítulo se monta de manera virtual una maqueta de la arquitectura de red propuesta. Una vez formada la maqueta de red se configura y se programan las diferentes aplicaciones necesarias para automatizar aspectos realacionados con las redes inalámbricas.
Con todo ello en marcha en la segunda parte de este capítulo se llevan a la práctica los tres casos de uso propuestos.

Capítulo 5 – Conclusiones

Una vez evaluados los tres casos de uso se obtienen unas conclusiones que se plasman en este capítulo.

2. Situación tecnológica

A continuación se va a describir la situación tecnológica de las diferentes herramientas que se pueden utilizar para llevar a cabo este proyecto.

2.1 Redes definidas por Software (SDN)

Las redes definidas por software son un enfoque de una arquitectura de red que permite que la red sea controlada mediante programación utilizando aplicaciones de software. SDN surge principalmente para solventar los problemas de las redes tradicionales, por ejemplo:

- CPDs que no pueden responder ante patrones de tráfico impredecibles, como picos de tráfico con demandas muy altas.
- Necesidad de configurar diferentes elementos de la red de manera individual ante cualquier tipo de configuración que se quiera desplegar a toda la red.
- Dificultad con el escalado de la red debido al aumento de complejidad de la propia red.
- Dependencia del fabricante ante nuevas capacidades que se deseen en los equipos.

El desarrollo de las SDN [1] empieza en 1990 cuando se empezó a incluir funciones programables en la red y se desarrollaron APIs para exponer recursos de los nodos de la red. Esto permitió reducir el coste computacional y avanzar en lenguajes de programación y tecnologías de virtualización.
A principios del 2000 empezó a surgir la idea de separar el plano de control y el plano de datos de los equipos de la red. Fueron las empresas de equipos hardware las que comenzaron a hacerlo mientras que los ISPs tenían el problema de la gestionar sus crecientes redes. Esto dio lugar a dos innovaciones, una interfaz abierta entre el plano de control y plano de datos y un control lógico centralizado en la red.

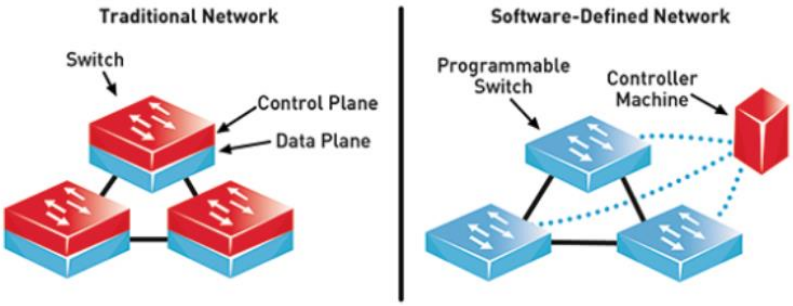


Figura 1 - Comparación red tradicional y red SDN.
Fuente: https://www.aicox.com/wp-content/uploads/2018/04/redes-definidas-por-Software.jpg?s=7005f33a67d736ff3bdf0c820b5e9de2e7094dbf

En la figura se puede ver a la izquierda una red tradicional, en la que los equipos cuentan tanto con plano de control como con plano de datos, por lo que los equipos son capaces de decidir qué hacer con el tráfico de red. En la

figura de la derecha se ve la arquitectura SDN, a los equipos se les elimina el plano de control y se centraliza en un equipo controlador. Desde ese momento los equipos para tomar decisiones con respecto al tráfico tienen que preguntar al controlador.

Para abordar la separación entre el plano de control y el plano de datos numerosos grupos de investigación empezaron a investigar nuevas arquitecturas para el control lógico centralizado.

2.1.1 OpenFlow

En la universidad de Stanford, un grupo de investigadores crearon el protocolo OpenFlow [2]. Las empresas adoptaron OpenFlow y permitieron acceso a sus APIs para que los programadores pudiesen controlar ciertos aspectos en el comportamiento de los equipos.

El protocolo OpenFlow permite la comunicación entre el controlador de la red y los elementos de red. En OpenFlow se centralizan las decisiones sobre los flujos de datos, de manera que la red se puede programar independientemente de los equipos que la componen, ya que cada switch tiene su software propietario.

OpenFlow funciona mediante tablas de flujos. Las tablas de flujos contienen información sobre los campos a comprobar dentro de los paquetes e instrucciones de qué hacer con ellos. Es el controlador el que modifica estas tablas instalando, modificando y borrando flujos.

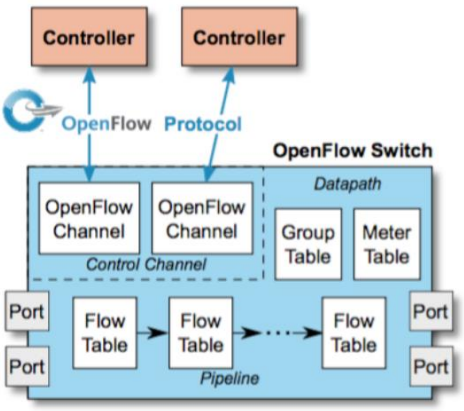


Figura 2 – Switch OpenFlow
Fuente: http://www.cables-solutions.com/wp-content/uploads/2018/07/what-is-openflow-switch.png

Como mínimo los equipos tienen una tabla de flujos, pero pueden ser más. El paquete a su llegada se procesa y pasa por las tablas de flujo (en orden y de una en una) para tomar la decisión de qué se hace con él. Si el paquete hace match en una regla se ejecuta la acción que se marque, y si no hace match en ninguna regla de ejecuta la acción por defecto (que pueden ser varias: enviar el paquete al controlador, hacer inundación con el paquete...). Para que un paquete haga match con una regla se pueden usar diversos campos, puerto de entrada, dirección MAC, dirección IP...
Si se decide re-enviar el paquete pasa por las tablas de salida.

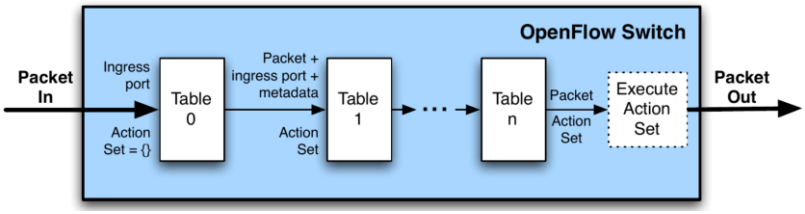


Figura 3 – Procesado de un paquete por un switch OpenFlow.
Fuente https://blog.zhaw.ch/icclab/files/2013/07/table-pipeline.png

2.1.2 Controlador SDN

El controlador SDN es el elemento principal de la arquitectura SDN, debido a que es el elemento que proporciona inteligencia a la red. Su función principal es la de manejar los flujos provenientes de los elementos de la red (switches, routers, firewalls, etc).
Los controladores tienen cuatro interfaces, norte, sur, este y oeste.
Las interfaces este y oeste se utilizan para la comunicación entre controladores, siempre y cuando la red tenga más de uno.
Las aplicaciones se comunican con el controlador por la interfaz norte a través de servicios, y estos servicios se encargan de procesar las solicitudes de una aplicación, ya sea instalar un flujo, solicitar información de la red o registrar eventos como los que envían los dispositivos al controlador cuando desconocen que acción se debe realizar con un paquete.
La comunicación entre el controlador y el resto de elementos de la red se hace a través de la interfaz sur y por lo general a través del protocolo OpenFlow.

Existe una gran variedad de controladores SDN de código abierto, de los cuales destacan los siguientes:

- NOX: Es el controlador original OpenFlow en el campo de SDN. Es ampliamente utilizado por grupos de investigación y ha servido como base para otros proyectos relacionados con SDN. No ha sido fuertemente implementado debido a deficiencias en su entorno de desarrollo.

- POX: Desarrollado a partir de NOX. Pensado también para usos dentro del entorno de investigación y educación. Tiene un entorno de desarrollo sencillo y dispone de un conjunto de varias APIs.

- Ryu: Escrito en Python, dispone de un conjunto de APIs bien definidas, las cuales permiten el desarrollo de nuevas aplicaciones de gestión de red. Cuenta a su vez con un conjunto de varios protocolos para la gestión de los dispositivos de red.

- Beacon: Es un controlador modular compatible con la programación basada en threads y eventos. Implementado en Java, ha servido como referencia para proyectos como Floodlight y OpenDaylight.

- Floodlight: Escrito en Java, cuenta con un sistema de módulos que lo hace extensible, fácil de configurar y de alto rendimiento.

- ONOS: Implementado en Java, se encuentra enfocado para casos de uso de proveedores de servicio con grandes redes WAN. Su arquitectura se centra en la tolerancia a fallos y la distribución de la red mediante múltiples controladores. Dispone de diversas interfaces norte y sur.

- OpenDaylight: Es un controlador multipropósito escrito en Java. Es ampliamente reconocido y dispone de un gran número de módulos que pueden ser utilizados según las necesidades de la organización. Incluye numerosas interfaces sur y una capa de abstracción por encima de esas interfaces.

Comparación entre controladores

De todos los controladores mencionados anteriormente se comparan tres de ellos, que son los que tienen una mayor relevancia. Los controladores comparados son ONOS, OpenDaylight (ODL) y Ryu.
La comparación está basada en los siguientes criterios: arquitectura, modularidad, escalabilidad, interfaces, telemetría, tolerancia a fallos, lenguaje de programación y comunidad.[3]

Arquitectura
ONOS y ODL cuentan con una arquitectura centralizada, por lo que son más fáciles de mantener, operar y tienden a ofrecer menos latencia tanto a la interfaz sur como a la norte. El problema que tienen es que a medida que la red crece, el controlador puede convertirse en el cuello de botella de la red. Ryu es diferente, tiene una serie de programas que componen su core que hacen que sea visto como una plataforma o un conjunto de herramientas sobre las que se puede desplegar la funcionalidad de controlador SDN.

Modularidad
La modularidad de los controladores depende principalmente de su lenguaje de programación y de su diseño. ONOS y ODL tienen mecanismos para conectar con módulos de código, usan contenedores para cargar paquetes en tiempo de ejecución, lo que les hace muy flexibles. Ryu está basado en Python y proporciona una API a los desarrolladores para cambiar la forma en que se gestionan y configuran los componentes.

Escalabilidad
Tanto ONOS como ODL permiten mantener un clúster y hacer failover si detectan que es necesario. Ryu no tiene la capacidad de mantener un clúster de manera intrínseca, sino que necesita usar herramientas externas para ello.

Interfaces
ONOS, ODL y Ryu tienen soporte para varios protocolos en la interfaz sur más allá de Openflow, como son P4 o Netconf. Con respecto a la interfaz norte, ONOS y ODL ofrecen el mayor número de interfaces con APIs gRPC y REST haciendo las integraciones más sencillas. Ryu sólo ofrece APIs REST, haciendo que sea más limitado.

Telemetría

Uno de los principales problemas con SDN es la extracción de telemetría para conocer el estado del sistema y ayudar a remediar problemas. ODL no tiene esta funcionalidad, de momento sólo tiene módulos experimentales. ONOS tiene módulos que permiten integrarlo con herramientas como Grafana o InfluxDB. Ryu no proporcionan ninguna funcionalidad de telemetría, es necesario usar herramientas externas.

Tolerancia a fallos

Como se ha comentado antes, ONOS y ODL permiten montar clústeres con varios controladores, en caso de fallo de un controlador habrá otro que sea seleccionado para controlar la red. Ryu necesita de herramientas externas para proporcionar esta funcionalidad, lo que tampoco es malo del todo, puesto que simplifica la arquitectura del controlador y le libera de mantener bases de datos distribuidas con información de estado. La alta disponibilidad se conseguiría por medio de ejecutar varias instancias iguales del controlador y controlar estas mediante un framework externo, el cual tendría que detectar y reiniciar los nodos en fallo.

Lenguaje de programación

ONOS y ODL están escritos en Java, por lo que los recursos a la hora de desarrollar son abundantes y, a parte, hay bastantes librerías disponibles que ayudan al desarrollo. Aun así, el usar Java implica que los procesos pueden ser pesados y requerir gestión de recursos para asegurar que tienen capacidad de respuesta. Ryu usa Python, un lenguaje con un gran soporte y una comunidad muy activa, pero el problema es que es un lenguaje interpretado por lo que al ser comparado con Java puede no ofrecer el mismo rendimiento.

Comunidad

ONOS y ODL tienen una comunidad de desarrolladores muy amplia bajo la fundación Linux. En este caso Ryu también es un controlador bien apoyado.

2.1.3 Redes SDN en industria

Las redes industriales sirven de interconexión para una gran variedad de dispositivos (sensores, PLCs, CNCs...) que llevan a cabo funciones de control, monitorización, etc. Estos sistemas generalmente se conectan a través de múltiples redes de datos, o cableadas o inalámbricas a las cuales se conectan les demandan nuevas prestaciones, de forma que el control y gestión de las redes deben estar acoplados a las condiciones de los propios sistemas. Es por tanto que existen requisitos como flexibilidad o mantenibilidad pero sin descuidar los requisitos de calidad de servicio. Las redes tradicionales no se adaptan a entornos dinámicos y las redes SDN ayudan a esta adaptación. Las redes industriales tienen determinados requisitos de latencia y disponibilidad, por lo que aprovechando la flexibilidad de la red y la programabilidad de la misma se permite la creación de políticas de tráfico en función de la prioridad del tráfico y del estado de la red. La configuración, asignación de recursos y la comunicación entre elementos es complicada. SDN juega un rol vital en la gestión de estas redes tan heterogéneas.

Los beneficios por tanto que ofrecen las redes SDN a las redes de dispositivos IoT son:

- Reducción de la complejidad de las redes tradicionales.
- Permitir un simple aprovisionamiento y gestión de recursos.
- Permite la centralización y abstracción del control y la automatización de funciones de red.
- Permite gestionar los flujos de los dispositivos dependiendo del tipo de dispositivo y flujo que se genera.
- Permite la división de la red, de manera que puedan aplicarse políticas de seguridad en función del tráfico de red, lo cual es muy interesante para securizar el tráfico industrial.

2.2 Virtualización de funciones de red (NFV)

Al proceso de crear particiones virtuales dentro de recursos físicos y que estas particiones virtuales emulen el mismo comportamiento del hardware físico, se le conoce como virtualización. La virtualización de usa en gran medida en el ámbito de la tecnología de la información, por ejemplo, a la hora de hacer divisiones en el disco duro o en memoria.
La virtualización de las funciones de red (NFV) es independiente de SDN, se podría llegar a implementar una VNF de manera independiente usando los paradigmas de orquestación y redes ya existentes. Sin embargo, si se combinan ambas, se mejora el aspecto de orquestación y gestión de la infraestructura NFV.
NFV consiste en trasladar las funciones de red de dispositivos dedicados a servidores virtuales. De esta manera se elimina la necesidad de adquirir hardware dedicado y también se eliminan las limitaciones asociadas a ese hardware, como por ejemplo baja escalabilidad o rendimiento insuficiente. Esta tecnología supone una gran ventaja con respecto a las redes tradicionales, ya que en caso de necesitarlo los servicios se proporcionan al momento y en el lugar donde se requieran, por lo que el uso de recursos está optimizado. Activar un servicio nuevo puede ser tan sencillo como encender una máquina virtual y apagando dicha máquina virtual se desactivaría.

La virtualización de funciones de red implementa funciones de red (VNFs) sobre una infraestructura para funciones virtualizadas (NFVI). Una VNF es cualquier función de la red que se virtualice y la NFVI es el conjunto hardware y software que forma el entorno donde se despliegan, ejecutan y gestionan las VNFs. Las VNFs están totalmente desacopladas del hardware que reside por debajo, de esa manera se asegura la independencia entre ambos.

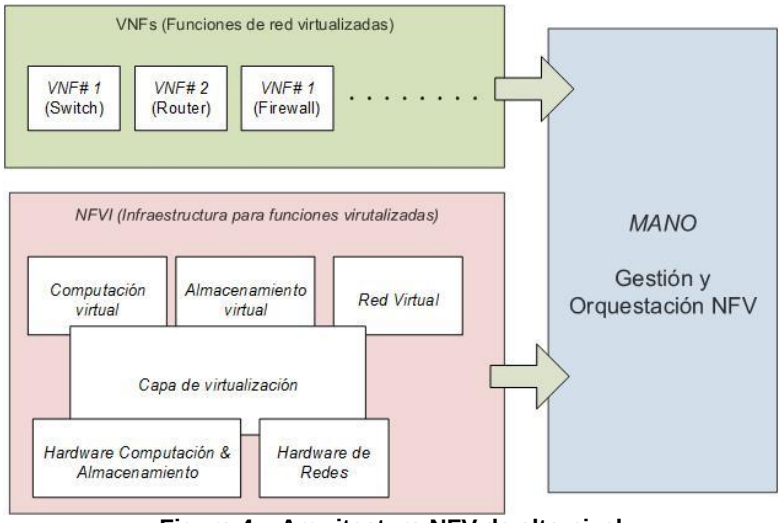


Figura 4 – Arquitectura NFV de alto nivel
Fuente: https://upload.wikimedia.org/wikipedia/commons/thumb/7/7b/Gr%C3%A1fico_arquitectura_NFV_-_JPG.jpg/600px-Gr%C3%A1fico_arquitectura_NFV_-_JPG.jpg

El bloque independiente MANO (Management and Orchestration) es el responsable del control de la arquitectura NFV e interactúa tanto con las VNFs como con la NFVI.
El bloque está compuesto por tres elementos, el orquestador NFV (NFVO), el gestor VNF (VNFM) y el gestor de la infraestructura virtual (VIM).

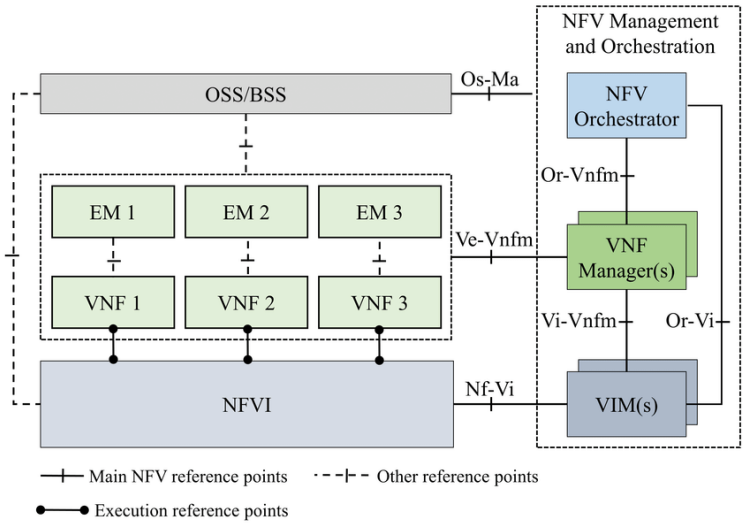


Figura 5 – Framework NFV
Fuente: https://www.researchgate.net/profile/Mohammad_Abu-Lebdeh/publication/320956787/figure/fig1/AS:558762048327680@1510230661335/ETSI-NFV-architectural-framework.png

Los administradores de red coordinan las VNFs a través del módulo NFVO. A parte, también se encarga de crear instancias de VNFM y de gestionar las solicitudes del módulo VIM.
El módulo VNFM se encarga de gestionar las VNFs, puede crear y eliminar instancias de VNFs, gestiona los fallos, la configuración y el rendimiento de las mismas.
El módulo VIM gestiona los recursos de la capa NFVI, creación y destrucción de máquinas virtuales y gestión de errores asociados a los recursos virtuales.

Tanto VNFI como MANO interactúan con el sistema de soporte de operaciones y negocio (OSS/BSS), el cual es el sistema en el que se manejan las operaciones y necesidades del dueño de la red, es decir, todo lo que tiene que ver con los requerimientos de administración, mantenimiento y configuración; así como también la interacción con el cliente final.

2.2.1 Herramientas de orquestación NFV

Dentro de este proyecto tiene especial importancia el bloque de gestión y orquestación, por lo que se va a realizar un estudio de diferentes herramientas a modo comparativo para ver cuál o cuáles serían las más adecuadas para este proyecto. También se explican algunos componentes de otros bloques que pueden llegar a usarse.
Distintos operadores y proveedores de servicios han desarrollado proyectos relacionados con el bloque MANO y estos proyectos se encuentran en constante actualización y evolución.

En septiembre de 2014 la fundación Linux anunciaron el proyecto OPNFV [4]. OPNFV facilita el desarrollo y evolución de componentes NFV dentro del ecosistema open source. A través de la integración de sistemas, el desarrollo y las pruebas OPNFV crea una plataforma de referencia NFV para acelerar la transformación de la empresa y de las redes de los proveedores de servicios.

OPNFV se centra en la construcción de infraestructura de NFV (NFVI) y gestión de infraestructura (VIM) mediante la integración de proyectos open source. La filosofía de OPNFV no es crear proyectos específicamente para sí mismo, sino en buscar proyectos de código abierto que tienen características que benefician su objetivo, e incorporarlos mediante un proceso de integración continua. OPNFV es una recopilación de una larga lista de proyectos, que se pueden clasificar en cuatro áreas: NFVI, MANO, VIM y Controlador SDN.
Esta es la arquitectura de OPNFV donde se pueden ver las colaboraciones con los proyectos open source que se comentaban.

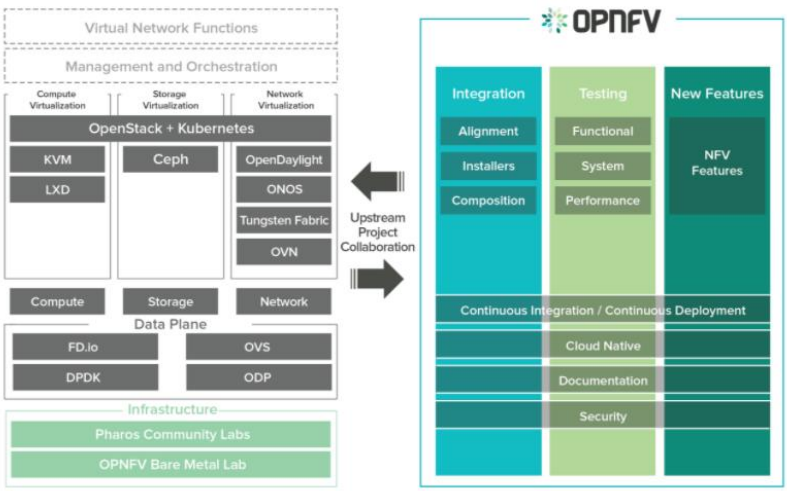


Figura 6 – OPNFV
Fuente
https://wiki.opnfv.org/download/attachments/2925118/opnfv_diagram_fraser_042318.png?version=1&modificationDate=1541999962000&api=v2

2.2.1.1 Tecnologías de Infraestructura de NFV (NFVI)

Máquina virtual basada en el kernel (KVM)

KVM [5] es una tecnología de virtualización de código abierto integrada en Linux. Con KVM se convierte un equipo en un hipervisor de nivel 1, lo que permite ejecutar múltiples máquinas virtuales. Cada máquina virtual se ejecuta como un proceso de Linux, con hardware virtual dedicado.

Docker

Docker [6] es una plataforma de software que permite el despliegue de aplicaciones de manera sencilla. Estas aplicaciones se empaquetan en contenedores que son autosuficientes, es decir, que incluyen todo lo necesario para que se ejecuten las aplicaciones. Los contenedores que se crean pueden ejecutarse en cualquier lugar, siempre y cuando se tenga instalado el software de Docker.

2.2.1.2 Tecnologías de gestión de la infraestructura (VIM)

OpenStack

Openstack [7] se desarrolló para proveer una plataforma encargada de orquestar la nube. Tiene la capacidad de desplegar máquinas virtuales, así como redes para interconectar estas máquinas, todo sobre infraestructura estándar. Para permitir todo está compuesto por un conjunto de componentes, de los cuáles estos son los principales:

Nova: Nova es una herramienta de gestión y acceso para los recursos físicos. Controla la planificación, creación y eliminación tanto de las máquinas virtuales como de todos los servicios necesarios.

Neutron: Neutron se utiliza para gestionar las redes. Permite la definición de redes que van a utilizarse en la topología para que todos los componentes estén interconectados.

Swift: Swift es el módulo encargado del almacenamiento de los archivos de sistema, asegurar su integridad y replicarlos por los diferentes discos con los que cuenta la infraestructura.

Cinder: Cinder proporciona almacenamiento persistente y permite el acceso al contenido alojado en las unidades de disco.

Keystone: Keystone se utiliza para autenticar y autorizar todos los servicios de Openstack, controla tanto el acceso de los usuarios a la plataforma como el acceso a servicios y aplicaciones.

Glance: Glance gestiona las imágenes (copias íntegras) de los discos duros de las máquinas desplegadas.

Horizon: Horizon es la herramienta que provee un portal web desde el que gestionar todos los servicios de Openstack.

Dentro de la arquitectura de Openstack hay tres nodos, el nodo controlador, el nodo de red y el nodo de cómputo. En cada uno de ellos se ejecutan distintos servicios. En el nodo controlador es donde se ejecutan la mayoría de los servicios y herramientas de Openstack. El nodo controlador suministra la API, tiene el panel web, el almacén de imágenes, el servicio de identidad, el servicio de Nova y Neutron. El nodo de la red proporciona servicios de red a las instancias de Nova usando Neutron y los servicios de DHCP. El nodo de cómputo es donde se instalan las instancias de máquinas virtuales.

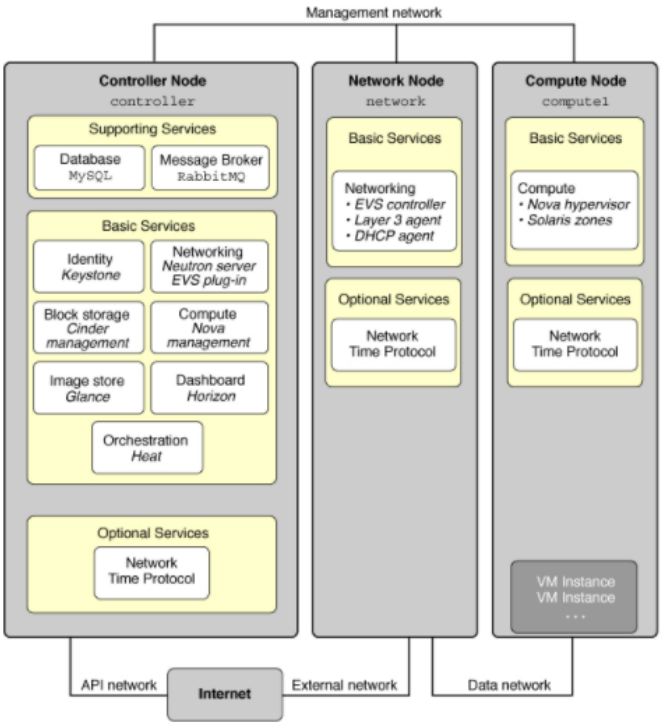


Figura 7 – Arquitectura de OpenStack

Fuente: https://docs.oracle.com/cd/E36784_01/html/E54155/figures/three-node-config.jpg

Kubernetes

Kubernetes es una plataforma de código abierto para administrar servicios. Ofrece un entorno de administración centrado en contenedores y orquesta la infraestructura de cómputo, redes y almacenamiento. Kubernetes está diseñado como una plataforma para poder construir un ecosistema de componentes y herramientas que hacen más fácil el desplegar, escalar y administrar aplicaciones. El plano de control de Kubernetes usa las mismas APIs que usan los desarrolladores por lo que los usuarios pueden escribir sus propios controladores, como por ejemplo un planificador. Este diseño ha permitido que otros sistemas sean construidos sobre Kubernetes.

Kubernetes no es un mero sistema de orquestación. De hecho, Kubernetes elimina la necesidad de orquestar. *Orquestación* se define como la ejecución de un flujo de trabajo definido: hacer A, luego B y entonces C. Kubernetes está

compuesto de un conjunto de procesos de control independientes y combinables entre sí que llevan el estado actual hacia el estado deseado. No debería importar demasiado cómo llegar de A a C. No se requiere control centralizado y, como resultado, el sistema es más fácil de usar, más poderoso, robusto, resiliente y extensible.

La arquitectura de Kubernetes está compuesta por un nodo maestro y por uno o varios nodos. El maestro es una colección de componentes que forman el panel de control de Kubernetes. Estos componentes se usan para todas las decisiones del clúster, lo que incluye la planificación y la respuesta de todos los eventos del clúster. Los nodos son hosts que son capaces de ejecutarse dentro de una máquina virtual.

2.2.1.3 Tecnologías de gestión y orquestación (MANO)

Open Orchestrator (Open-O) / Open Network Automation Platform (ONAP)

Open-O [8] se centra principalmente en permitir la agilidad de extremo a extremo a través de múltiples dominios utilizando una plataforma para la orquestación de NFV y SDN.

Entre los beneficios de Open-O están:

- Permitir a los operadores capitalizar la arquitectura de NFV y SDN.
- Permitir la composición de servicios multi-dominio, multi-ubicación y de extremo a extremo.
- Adopción de un modelo de información común para toda la industria para garantizar la extensibilidad, la eficiencia en la implementación y la interoperabilidad.
- Proporcionar una plataforma común para los proveedores de VNF, simplificando y acelerando la incorporación, el desarrollo y el despliegue de VNF.
- Mejorar el ciclo de vida general del servicio, a través de la automatización impulsada por el modelo para permitir la reutilización y las actualizaciones incrementales.
- Proporcionar abstracción para operar sobre redes SDN, NFV y redes tradicionales.

OpenMANO

OpenMANO [9] es un proyecto de código abierto que proporciona una implementación práctica de la arquitectura de referencia de gestión y orquestación NFV. OpenMANO proporciona tres módulos de software:

- OpenMANO: Es la implementación de un orquestador de funciones virtualizadas de red, que permite la creación de escenarios de red complejos. El módulo tiene una interfaz con la capa VIM a través de su API y ofrece una interfaz norte basada en REST a través de las que se ofrecen los servicios de NFV.

- Openvim: Es la implementación de la capa VIM. El módulo tiene una interfaz con los nodos de computación de la infraestructura NFV y con el controlador openflow para proporcionar capacidades de computación y de enrutado así como para desplegar máquinas virtuales. Ofrece también una interfaz norte, donde se ofrecen servicios cloud mejorados en los que se incluye la creación, borrado y gestión de imágenes, instancias y redes.

- OpenMANO-gui: web interfaz gráfica de usuario que interacciona con la API de openmano de manera amigable.

Juju

Juju [10] es una herramienta de despliegue de servicios y orquestación. Los desarrolladores son capaces de crear servicios de manera independiente, y hacer que estos servicios se comuniquen a través de un protocolo de configuración simple. Los usuarios pueden implementar estos servicios en un entorno, asemejándose a la forma cómo las personas son capaces de instalar una red de paquetes con un gestor de paquetes. Juju es un software de código libre y permite desplegar, configurar, gestionar y mantener servicios con agilidad.
Los desarrolladores son capaces de crear servicios, llamados charms y hacer que estos servicios coordinen su comunicación y configuración a través de un protocolo sencillo, de esta forma un charm inidica a juju cómo gestionar un servicio.
Los charms están compuestos de metadatos, datos de configuración, y ganchos (hooks) con algunos archivos de soporte adicionales. Charms utilizan hooks, normalmente escritas como scripts, los cuales juju invoca para gestionar el ciclo de vida del servicio. Hay dos tipos de hooks: Hooks de unidad y Hooks de relación. Los primeros incluyen instalar, configurar, iniciar, actualizar-charm, y detener. Los segundos, hooks de relación, incluyen juntar, cambiar, desplegar, y romper relaciones. De la misma forma que toda plataforma moderna de gestión, juju también puede ser utilizado junto con algún orquestador.

RIFT.ware

Rift.ware [11] es una plataforma de NFV de código libre que simplifica el desarrollo, despliegue y gestión de servicios de red virtuales.
Los beneficios de Rift.ware son los siguientes:

- Simple: RIFT.ware abstrae la complejidad de la red subyacente y la infraestructura de computación de la nube para reducir el desarrollo, la integración, costos y el esfuerzo de integración. Presenta la capacidad de que constructores VNF puedan migrar de los equipos dedicados hacia servicios virtualizados en poco tiempo.

- Seguro: Funciones de red pueden aprovechar las ventajas de colocar sobre plataformas confiables la carga de trabajo segura y así generar

soporte en cifrado para obtener un nivel de seguridad dentro de las empresas.

- Escalable: RIFT.ware permite la posibilidad de iniciar con una única VNF en una máquina virtual y a continuación escalar a miles de máquinas virtuales en múltiples sitios y nubes híbridas con capacidad de recuperación de nubes a gran escala.

- Código Abierto: Rift.ware es una plataforma de código libre que evita la dependencia de un proveedor, facilita interoperabilidad, así como proporcionar opciones y flexibilidad.

La Plataforma se diseñó bajo las normas establecidas por ETSI y está construida con las más modernas tecnologías web, con el fin de ofrecer automatización, capacidades de hiperescalabilidad y servicios de red virtualizados. RIFT.ware engloba virtualización de cómputo y NFVs para implementar dentro de cualquier organización y establecer comunicaciones en cualquier infraestructura de nube a toda escala.

Los componentes de RIFT.ware son:

- Gestor del ciclo de vida: La capa de gestión del ciclo de vida forma los recursos informáticos y redes virtuales dentro de una topología estructurada, que puede soportar varias VNFs.

- Gestor VNF: Supervisa el ciclo de vida de las instancias VNF (Iniciado, escalado y monitorización)

- Orquestador NFV: Administra el ciclo de vida de los servicios de red y orquesta los recursos a través de múltiples gestores de infraestructuras virtualizadas.

2.3 Tecnología de acceso inalámbrico a la red (IEEE 802.11)

2.3.1 Introducción

El estándar 802.11 [12] es miembro de la familia 802, que se corresponde a las tecnologías para redes de área local. En concreto, IEEE 802.11 es el estándar para redes inalámbricas. El estándar define las dos capas más bajas del modelo de referencia, la capa de acceso al medio y la capa física.

La capa de control de acceso al medio se encarga del control del medio de transmisión, ya que es compartido por múltiples dispositivos que tienen la posibilidad de transmitir simultáneamente interfiriendo unos con otros. Se hace uso de protocolos como CSMA (Carrier Sense Multiple Access) cuyo objetivo es detectar si el canal está inactivo, permitiendo emitir si lo está. Para evitar colisiones se hace uso de protocolos como CA (Collision Avoidance) junto con CSMA, y su función es la de escuchar el canal repetidas veces para detectar si está libre de otras transmisiones.

La capa física define las especificaciones eléctricas y físicas para los dispositivos, particularmente define la relación entre los dispositivos y los medios de transmisión. Las principales funciones de la capa son el establecimiento y la finalización de las conexiones con el medio, participación en la compartición de recursos entre múltiples usuarios (control de flujo) y modulación de los datos y señales transmitidas por el canal de comunicaciones.

Se utilizan dos bandas frecuenciales, la de 2.4 GHz y la de 5 GHz. Estas bandas de frecuencias están divididas en trozos de un determinado ancho frecuencial que se llaman canales y los puntos de acceso WiFi (APs) son capaces de utilizar cualquiera de estos canales.
Las diferencias entre ambas bandas principalmente son:

	2.4 GHz	5 GHz
Número canales	14 canales (1-13)	25 canales (36-64 y 100-140)
Alcance	Mayor alcance	Menor alcance
Ancho de banda	54 Mbps	11 Gbps (teórico, en el nuevo estándar 802.11ax)
Interferencias	Mayor número, debido a que hay un número muy grande de dispositivos que la utilizan.	Menor número, debido a que la banda está menos utilizada.

Tabla 1 – Comparación entre las bandas de 2.4GHz y 5GHz

El estándar está en continua evolución y a día de hoy existen varias versiones, de las cuales las más reconocidas son estas:

Versión	Año	Banda de frecuencia (GHz)	Ancho de canal (MHz)	RF	Radio	Velocidad máxima
802.11b (WiFi 1)	1999	2.4	20	HR-DSSS	SISO	11 Mbits/s
802.11a (WiFi 2)	1999	5	20	OFDM	SISO	54 Mbits/s
802.11g (WiFi 3)	2003	2.4	20	OFDM	SISO	542 Mbits/s
802.11n (WiFi 4)	2007	2.4 y 5	20 y 40	OFDM	MIMO	600 Mbits/s
802.11ac (WiFi 5)	2014	5	20, 40, 80 y 160	OFDM	MU-MIMO	6.93 Gbits/s
802.11ax (WiFi 6)	2019	2.4 y 5	20, 40, 80 y 160	OFDM y OFDMA	MU-MIMO	11 Gbits/s

Tabla 2 – Comparación entre versiones de 802.11

2.3.2 IEEE 802.11ax

El estándar 802.11ax (WiFi 6) se ha ratificado en 2019 y se trata de la sexta generación de WiFi. El estándar se basa en las fortalezas de 802.11ac,

agregando eficiencia, flexibilidad y escalabilidad que permite a las redes nuevas y existentes aumentar la velocidad y la capacidad con las nuevas aplicaciones.
Las formas en las que WiFi 6 mejora la experiencia de usuario son [13]:

Soporte de múltiples usuarios con OFDMA [14] (acceso múltiple por división de frecuencias ortogonales). OFDMA se utiliza para conseguir que un conjunto de usuarios puedan compartir el espectro para aplicaciones de baja latencia. Se divide el canal en un conjunto de subportadoras y se permite a los usuarios transmitir en diferentes subportadoras. Todas las subportadoras se dividen en dominios de frecuencia cada uno de los cuales se llama subcanal.
En el mismo segmento de tiempo múltiples usuarios pueden transmitir simultáneamente a través de diferentes subcanales. Con las versiones anteriores de 802.11 sólo podía transmitir un dispositivo simultáneamente. Este aspecto favorece a los dispositivos IoT debido a que son muy numerosos y si están cercanos pueden transmitir simultáneamente cada uno por un subcanal diferente.

Soporte de múltiples usuarios con MIMO (múltiples entradas, múltiples salidas). MU-MIMO se introdujo en 802.11ac y permite a un punto de acceso comunicarse con varios usuarios de manera simultánea. Esto reduce el tiempo de espera de los dispositivos conectados a la hora de recibir la señal e incrementa la velocidad de la red. Un punto de acceso puede transmitir 8 flujos simultáneamente. Esto favorece a los dispositivos IoT en el sentido de que si hay un dispositivo que quiere transmitir tiene que estar un menor tiempo a la espera, esto es crucial para reducir el consumo del dispositivo. Si el consumo baja la batería puede ser menor y el dispositivo más pequeño, aspecto deseable en ciertos casos.

Reutilización del espacio (BSS Coloring) [15]. Un BSS está compuesto por un un punto de acceso con uno o más clientes. En WiFi 6 se puede diferenciar entre BSS por medio de asignaciones de colores. Cada punto de acceso y su celda tienen un color y el comportamiento es diferente si los puntos de acceso cercanos tienen el mismo color o un color diferente. Si el color no es el mismo los puntos de acceso saben que no van a producirse interferencias. Si el color es el mismo se considera que se podrían causar interferencias y la BSS puede cambiarse de color. Con esta funcionalidad paquetes de diferentes dispositivos pueden estar en el mismo canal al mismo tiempo.
Este aspecto puede ayudar a los dispositivos IoT debido a que las ventajas que oferta se acentúan en entornos de red muy densos como los empresariales con gran cantidad de APs.

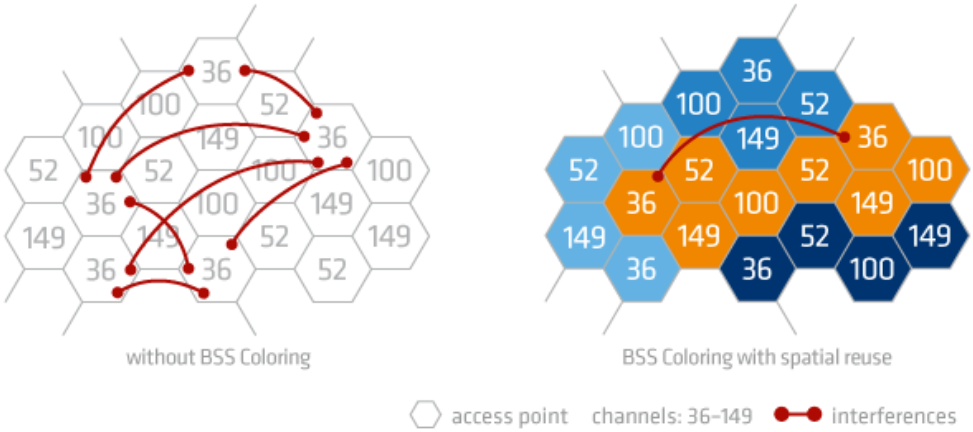


Figura 8 – BSS Coloring
Fuente: https://www.lancom-systems.com/fileadmin/images/technologie/Wi-Fi_6/Wi-Fi-6-BSS-Coloring_EN.png

Alto rendimiento con modulaciones de mayor orden. El anterior estándar WiFi está limitado a una modulación 256 QAM, mientras que WiFi 6 utiliza una modulación 1024 QAM. QAM es un esquema de modulación a través del cual los datos son transmitidos sobre radiofrecuencias. Dos portadoras desfasadas 90 grados son moduladas y la salida resultante consiste en variaciones de fase y amplitud. Cada una de estas variaciones viene representada por un punto dentro del diagrama de constelación del esquema de modulación y estos puntos representan símbolos. Cada símbolo de la constelación codifica un mayor número de bits de datos. La modulación 1024 QAM frente a 256 QAM consigue mejorar la velocidad de transmisión de datos en un 25% bajo buenas consiciones.

Eficiencia de transmisión. WiFi 6 mejora la eficiciencia en dos aspectos, en el tiempo de espera de los clientes y permitiendo la conexión a clientes con chips que solo soportan canales de 20MHz.
En anteriores generaciones de WiFi los clientes se mantenían a la espera unos milisegundos (el intervalo de las tramas beacon), pero era necesario que los dispositivos estuviesen controlando si había o no transmisiones de datos. Esto puede ser problemático para dispositivos IoT y para dispositivos con poca batería. Ahora los clientes pueden negociar con el punto de acceso el tiempo que van a estar a la espera.
Por otro lado, se permite la conexión de dispositivos que tienen chips sólo compatibles con canales de 20MHz cuando antiguamente debían soportar canales de 20, 40 y 80MHz. Esto permite el crear chips más pequeños, baratos y con menos requerimentos a nivel de batería. Como se ha comentado anteriormente esto beneficia a los dispositivos IoT debido a que si los chips tienen menos requerimentos de batería el tamaño de la batería es menor y por tanto los dispositivos IoT pueden ser más pequeños. Esto es deseable muchas veces, sobre todo en sensores.

El WiFi 6 y las redes móviles de última generación (5G) son tecnologías complementarias a pesar de tener diferentes usos. Sin embargo, las redes WiFi por lo general son redes que no son de dominio público, lo que las hacen más interesantes para uso empresarial.

En las empresas es necesario dotar de conectividad de red a todos los elementos pero en la mayoría de los casos se debe garantizar que tengan tanto acceso a aspectos de la red interna de la empresa como a elementos en internet. Es en este punto donde se hace conveniente poder controlar el elemento que proporciona acceso a la red, de tal manera que puedan controlarse aspectos de seguridad relativos a la red (segmentación de redes en VLANs, creación de políticas específicas en firewalls, etc..).
Esto se consigue mejor teniendo un equipo controlador on-premise, es decir mediante WiFi con un controlador, en este caso una función virtualizada de red. A parte, tal y como se ha explicado WiFi 6 provee de varias ventajas frente al anterior estándar, sobre todo ventajas relativas a dispositivos IoT, los más numerosos.

2.3.3 Formato de las tramas de 802.11

El estándar 802.11 define varios tipos de tramas que pueden clasificarse en tramas de datos, de gestión y de control. Las tramas de datos son las que transportan la información. Las tramas de gestión son las que permiten mantener la comunicación entre estaciones inalámbricas (autenticación, asociación, desasociación, tramas beacon, etc). Las tramas de control son las tramas que se utilizan para controlar el acceso al medio.

El formato general de las tramas [16] es el siguiente:

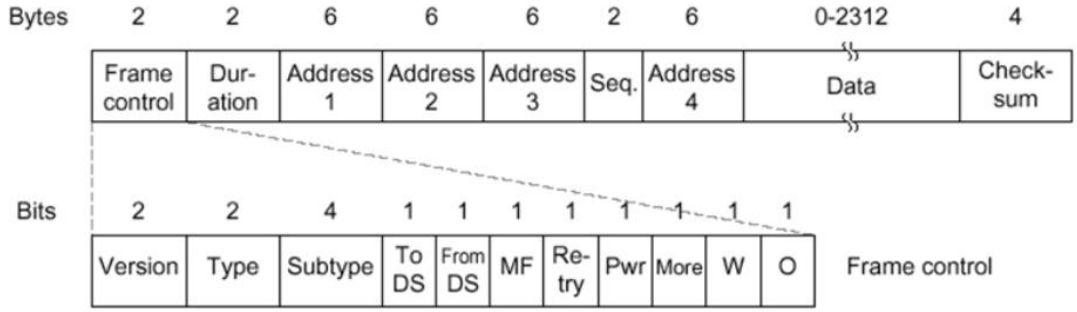


Figura 9 – Formato de las tramas WiFi
Fuente https://www.monografias.com/trabajos107/lans-inalambricas-ieee-802-11-y-802-15/Diapositiva37.png

Dentro de la trama de control hay varios campos:

- Version: Versión del protocolo
- Type and Subtype: Identifican la función de la trama, control, datos o gestión.
  To DS: Tiene el valor 1 para tramas de datos destinadas al sistema de distribución.
- From DS: Tiene el valor 1 para las tramas que salen desde el sistema de distribución.
- MF: Tiene el valor 1 para las tramas que están divididas en más fragmentos.
- Retry: Tiene el valor 1 si la trama es una retransmisión.
- PWR: Tiene el valor 1 si el cliente está en modo ahorro de energía.

- More: Tiene el valor 1 para indicarle a un cliente que hay tramas en búfer.
- W: Tiene el valor 1 si la trama contiene información encriptada con WEP.
- O: Tiene el valor 1 si la trama no necesita ser reordenada.

El resto de campos de la trama:

- Duration: Tiempo que se requiere para transmitir la trama desde la estación.
- Address 1: Dirección MAC del nodo destino final de la trama
- Address 2: Dirección MAC del nodo origen de la trama
- Address 3: Dirección MAC del destinatario inmediato de la trama
- Seq.: Número de cada fragmento de la trama
- Address 4: Dirección MAC del dispositivo inalámbrico que transmitió la trama
- Data: Información que transporta, por lo general un paquete IP
- Checksum: Contiene un campo de comprobación de la trama.

Es necesario el conocimiento de estas tramas y sus campos para ver como se adaptan en una arquitectura de red de SDN con NFV.

2.3.4 Arquitectura de redes WiFi

En cuanto a arquitectura las redes inalámbricas son más bien simples, hay dos vertientes, la vertiente doméstica/pequeña oficina y la vertiente empresarial. Las redes WiFi de pequeñas oficinas tienen una arquitectura muy sencilla, se componen simplemente de un AP que proporciona acceso a la red y a internet.

La arquitectura de redes WiFi empresariales es más compleja debido a que necesitan un mayor número de APs. En el caso de hacer un despliegue de red con varios APs es necesario colocarlos estratégicamente, de manera que no se produzcan interferencias entre APs cercanos. En el caso de tener dos APs próximos es muy posible que el radio de cobertura de uno se solape con el radio del otro, y en el caso de utilizar el mismo canal se producen solapes e interferencias.
Para controlar estos aspectos en las redes empresariales se utiliza un equipo controlador de toda la infraestructura WiFi (WLC). Proveedores como Cisco [17] o HP [18] proporcionan este tipo de controladores para controlar infraestructuras inalámbricas. Los APs al conectarse por primera vez a la red se conectan con el WLC y mediante el protocolo de aprovisionamiento de APs (CAPWAP [19]) se descargan la configuración que tienen que instalar.
Por lo general el tráfico de los APs se tuneliza hasta el WLC y es este equipo el encargado de tomar decisiones relativas al tráfico, aunque es posible configurar los APs para que sean capaces de gestionar el tráfico sin mandarlo hasta el WLC. Esta configuración se utiliza en los escenarios en los que los APs no están cercanos al WLC, por ejemplo en el caso de que el WLC esté en la sede principal de una empresa y se disponga de APs en sedes secundarias.

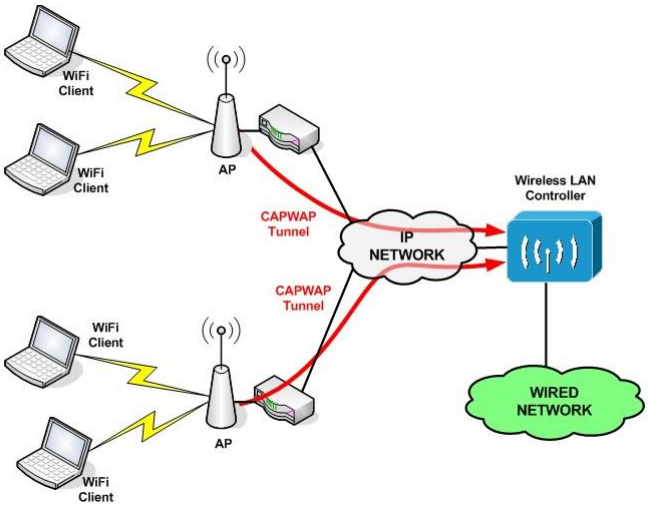


Figura 10 – Arquitectura WiFi empresarial
Fuente: https://www.networkstraining.com/wp-content/uploads/2015/12/centralized-wireless-architecture.jpg

Dentro de esta arquitectura de red, se aprecia que el elemento que puede sustituirse por una función de red virtual es el WLC. Es especialmente interesante la virtualización del WLC debido a que al hacerlo se hace posible que una empresa con varias sedes cuente con su propio WLC. Esto reduciría en gran medida las latencias entre APs y el WLC al que están asociados consiguiendo una mejor experiencia para el usuario final.

Como se puede ver, las redes inalámbricas desplegadas en empresas tienen una arquitectura muy similar a las redes SDN, siendo el equipo WLC el que hace las veces de controlador de todos los APs.

2.3.5 Redes WiFi con SDN (SD-WLAN)

Uno de los retos de las redes WiFi es conseguir una gestión más eficiente del ecosistema de redes inalámbricas, lo cual hace que sea necesaria una gran simplificación en la gestión de la red. Con SDN sobre la infraestructura inalámbrica se puede conseguir esa simplificación, pero es necesario definir la interacción de los dispositivos WiFi con la red SDN y adaptar todas las funcionalidades que se ofrecen en las redes tradicionales a VNFs.
Las arquitecturas basadas en controlador ofrecen las siguientes funcionalidades entre otras:

- Seguridad mejorada
- Ahorro de energía para dispositivos
- Posibilidad de aplicar políticas de QoS
- Tunelización de información
- Alta velocidad en el proceso de handover
- Balanceo de carga

Hay varias soluciones que persiguen integrar las infraestructuras de red inalámbricas dentro de redes SDN.

Chandelle [20]

Esta aproximación consiste en utilizar SDN por encima de la infraestructura inalámbrica. Hay varios elementos dentro de la red, el controlador SDN, el controlador WLC, switches OpenFlow y APs WiFi.
El controlador SDN controla mediante OpenFlow los switches, de la misma manera que lo hace tradicionalmente.
El equipo WLC es un equipo más en la red y como tal es capaz de conectarse con los APs permitiendo el uso de CAPWAP. A parte, el equipo WLC es capaz de comunicarse con el controlador SDN a través de su interfaz norte y esto lo hace para indicarle al controlador la necesidad de instalar nuevos flujos en los switches ante la movilidad de los clientes a través de los APs. Esto consigue que el handover sea más fluido. Si un cliente decide moverse, el AP con el que está asociado manda una notificación al equipo WLC avisando de nivel bajo de señal. El equipo WLC le manda al cliente una recomendación de otro AP y si el cliente se conecta con ese AP, el equipo WLC recibe un mensaje de dicho AP para autenticar al cliente. Una vez autenticado el cliente inicia el four-way handshake para establecer la conexión y es aquí cuando el equipo WLC advierte al controlador SDN para que instale los nuevos flujos en los switches.

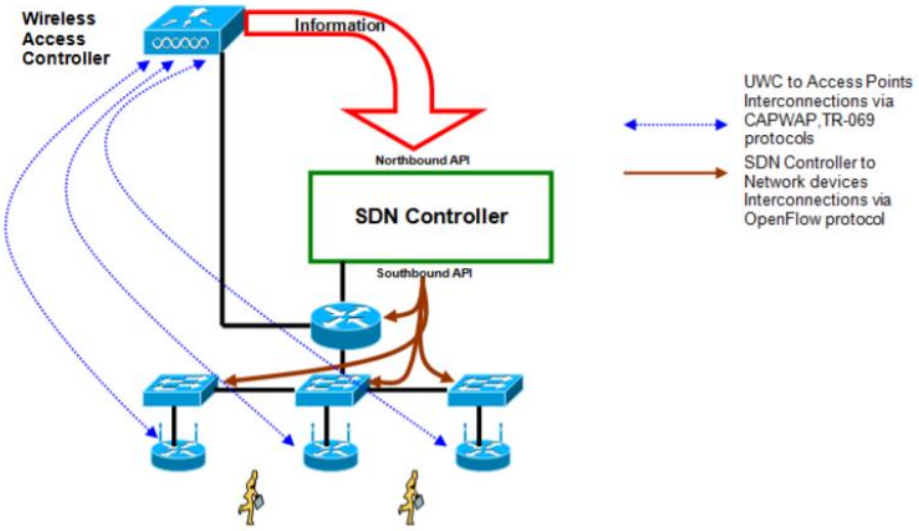


Figura 11 – Arquitectura Chandelle
Fuente:
https://www.researchgate.net/profile/Ruslan_Smeliansky/publication/322764684/figure/fig2/AS:667820933279763@1536232326652/The-basic-scheme-of-integrating-CAPWAP-wireless-controller-with-SDN-OpenFlow-controller.ppm

Odin

Odin [21] es un framework SDN para la gestión de infraestructuras inalámbricas. Integra servicios como gestión de la red, AAA (autenticación, autorización y auditoría), movilidad, balanceo de carga, ciber seguridad, etc.
Los componentes de la arquitectura de Odin son:

- Light Virtual AP (LVAP): Un sólo AP es capaz de estar dividido en varios APs de manera virtual. LVAP representa el enlace entre los clientes y los

APs virtuales. LVAP contiene la dirección MAC de la estación, la dirección IP, el SSID de la red y el BSSID del AP. LVAP hace pensar al cliente que es el único conectado al AP, de manera que la comunicación es unicast. Además, es posible mover el LVAP de un AP a otro, consiguiendo movilidad sin desconectar al cliente de la red.

- Maestro Odin: Es la aplicación por encima del controlador. Se implementa sobre el controlador Floodlight. Permite crear, añadir o eliminar LVAP, pedir estadísticas a los APs, actualizar tablas de flujos, etc.
- Agente Odin: Es la aplicación sobre el AP físico. Puede procesar LVAP y almacenar información acerca de las estaciones que tiene conectadas. Este agente captura tramas de los clientes y si para un cliente concreto no hay un LVAP creado se le solicita al maestro. También se encarga del peroceso de autenticación y asociación. La autenticación se realiza mediante el almacenamiento de una clave en el LVAP. Una vez hecha la asociación el agente indica dónde se le ha proporcionado acceso a la red al cliente.

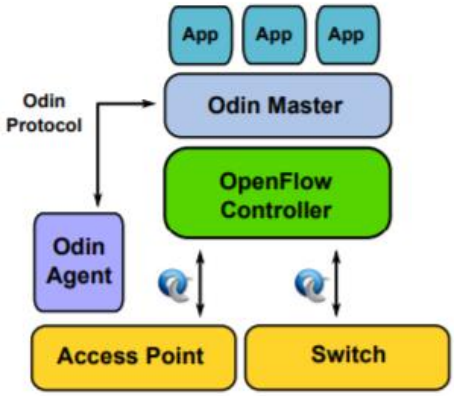


Figura 12 – Arquitectura de Odin
Fuente: https://www.researchgate.net/profile/Raghunath_Deshpande/publication/282322426/figure/fig7/AS:670484299534359@1536867322632/shows-AeroFlux-architecture-9-It-contains-four-architectural-building-blocks-1.ppm

Soluciones privadas

Los fabricantes de electrónica de red ofrecen soluciones de SDN que se aplican a arquitecturas inalámbricas.
Cisco ofrece dos variantes, su solución propia (Cisco enterprise NFV [22]) que consiste en virtualizar varios de sus appliance físicos. Los appliance que cisco ofrece virtualizar son routers (ISRv), firewalls (ASAv, NGFW), aceleradores WAN (vWAAS) y controladora WiFi (vWLC). La virtualización de estos appliances permite simplificar el despliegue de elementos de red basados en la necesidad de negocio, comparado con el despliegue de elementos físicos.
Para el caso de redes WiFi, Cisco ofrece su equipo WLC como appliance físico, todos los APs se configuran en modo CAPWAP y se asocian con el WLC. A parte, desde ese equipo se despliegan todas las configuraciones de red a todos los APs o a grupos concretos de APs (WLANs, políticas de seguridad y QoS, etc). El virtualizar este equipo como función de red sólo implica que no hay un

equipo físico en la red, debido a que aunque sea una máquina virtual tiene una IP y los APs se conectan con una IP.

La otra variante que ofrece Cisco es la solución Meraki, basada en un controlador en la nube. Este controlador en la nube se utiliza para controlar todo el ecosistema Meraki, tanto switches como APs. Esto no es una solución SDN como tal porque los switches tienen plano de control embebido, pero los equipos tienen dependencia del controlador en la nube, sobre todo para el despliegue de configuración.

Otro de los grandes fabricantes que ofrece una solución SDN que se integra con redes inalámbricas es HP con sus equipos Aruba[23]. Aruba ofrece la posibilidad de que sus APs trabajen de manera autónoma y con un controlador que puede estar tanto on-premise (appliance físico) como en la nube. Independientemente del modo de funcionamiento de los APs pueden interactuar mediante OpenFlow con un controlador SDN. Cada AP ofrece el acceso a la red WiFi como puertos lógicos. Cuando un cliente se conecta a uno de esos puertos envía la información a través de OpenFlow al controlador, de manera que el controlador puede enviar los flujos dirigidos al cliente al AP correcto.

Fortigate, que ha adquirido Meru Networks, ofrece una solución para aprovisionamiento de dispositivos WiFi que se integra con SDN.

Todas las soluciones propietarias ofrecen los mismas posibilidades que los equipos físicos.

2.4 Trabajos relacionados

Hay varios trabajos de otros autores que tratan temas relacionados con el tema del presente trabajo, por lo que en este apartado se pretende explorar algunos de esos trabajos. El objetivo de hacer esto es conocer el estado del arte del tema del que trata este trabajo.

En el trabajo “Proramming the Enterprise WLAN: An SDN Approach, Autor: Lalith Suresh Puthalath, Año: 2012” [24] se expone la necesidad de introducir programabilidad dentro de las redes inalámbricas empresariales. Para ello, se utiliza el framework Odin, basado en SDN. Se pueden construir aplicaciones encima de Odin, y con esto se consigue virtualizar los servicios de una red inalámbrica. Ejemplos de aplicaciones que se pueden diseñar son: movilidad de clientes, balanceo de carga o control de acceso. En el trabajo se explora el uso de la aplicación para permitir la movilidad de clientes entre APs.

El trabajo “SDN sobre redes 802.11: Simulación mediante Mininet-WiFi, Autor: Eduardo Gregorio Sáinz, Año: 2016” explora la aplicación de SDN a redes inalámbricas y el uso de OpenFlow en la interfaz sur del controlador. Se realizan varias simulaciones a través de la herramienta Mininet. Esta herramienta permite la creación de escenarios de red, formados por un controlador SDN, varios switches OpenFlow (Open vSwitch) y hosts. En este

caso, la extensión Mininet-WiFi permite incluir APs y realizar simulaciones de la movilidad de los clientes.

El trabajo “Chandelle: Smooth and Fast WiFi Roaming with SDN/OpenFlow, Autores: Sergey Monin, Alexander Shalimov, Ruslan Smeliansky, Año:2018” presenta el sistema Chandelle. La propuesta que realizan consiste en el uso de un sistema controlador que realiza el aprovisionamiento de APs y que se comunica con el controlador SDN para mejorar la eficiencia del roamming de clientes. Se centran principalmente en el problema de movilidad en las redes tradicionales.

Por último, el trabajo “High Performance SDN WLAN Architecture, Autores: Kristian Kostal, Rastilav Bencel, Michal Ries, Peter Truchly e Ivan Kotuliak, Año: 2019” se centra en el diseño de una arquitectura de red inalámbrica basada en SDN. Se propone una arquitectura de red que se centra en cumplir los objetivos de proveer un tiempo de roamming mínimo y mejorar la seguridad inalámbrica. Su propuesta se basa en el uso de una aplicación que se comunica vía API con el controlador de la red.

En el presente trabajo se presenta una arquitectura de red inalámbrica que se pretende que sea lo más eficiente posible. Se considera que la opción más eficiente es la del uso de una controladora para el aprovisionamiento de APs y dejar la parte de gestión de tráfico a los switches por lo que se adopta una solución similar a la propuesta en el trabajo Chandelle.
La diferencia con este trabajo es que Chandelle está orientado a la mejora del roaming entre clientes y este trabajo se centra en la gestión del tráfico inalámbrico como tal. Además, se propone el uso de aplicaciones por encima del controlador que sirven para convertir los equipos OpenFlow en equipos de red especializados (como firewalls por ejemplo). Otra diferencia adicional es que la controladora WiFi, al ser un equipo cuya función es la de aprovisionar APs y clientes no necesita de comunicación con el controlador por la interfaz norte, por lo que se simplifica la solución.
Por último, la parte que más diferencia este trabajo con el resto es la propuesta de una controladora WiFi virtual, lo que tiene una gran cantidad de ventajas. Una de esas ventajas es la posibilidad de poder realizar el despliegue de tantas controladoras como sea necesario y el apagado de las mismas en un tiempo muy bajo y con muy poco esfuerzo.

2.5 Resumen de puntos clave

Del estudio de la situación tecnológica por separado de varias tecnologías se extrae que para poder diseñar una arquitectura adecuada para integrar redes SDN con una infraestructura inalámbrica hacen falta varios elementos dentro de las tres capas de SDN.

Capa de aplicación:

Es necesario el diseño de aplicaciones que se comuniquen con el controlador de la red SDN. Para tener una integración total entre SDN y una red inalámbrica las aplicaciones tienen que suplir todas las funciones de red que se

ofrecen en una red inalámbrica. Estas funciones son autenticación de clientes, movilidad de clientes, seguridad, administración y gestión de los recursos inalámbricos.

Capa de control:

El elemento más importante dentro de la red es el controlador SDN. Tiene dos aspectos muy importantes, la interfaz norte y la interfaz sur. Cuanto mayor sea el número de protocolos que soporta en ambas interfaces mayor facilidad se tendrá a la hora de integrarlo en la red. Es deseable que en la interfaz norte se ofrezca una API Rest a través de la cual obtener información de la red e instalar flujos en el controlador. En la interfaz sur, a parte del protocolo OpenFlow es deseable que se permita el uso de protocolos como SNMP o NETCONF para configurar elementos de la red.

Capa de infraestructura:

Es necesario el uso de electrónica de red que se comunique con el controlador SDN por su interfaz sur. El controlador es consciente de todos los elementos de red, las conexiones entre ellos y es capaz de instalar flujos en las tablas de los equipos. Las aplicaciones de la capa de aplicación indican al controlador qué flujos debe instalar en los equipos, por lo que a través de las aplicaciones se puede conseguir simular cualquier tipo de comportamiento en la red, por ejemplo control de acceso o establecimiento de caminos redundantes. La electrónica de red debe soportar tanto el acceso de manera cableada o inalámbrica.

A parte de los elementos SDN es necesario:

- Llevar a cabo la virtualización de funciones de red que cumplen los elementos de red que actualmente tienen una función dedicada.

- Una herramienta cuya función sea la de virtualizar elementos de la infraestructura de red.

3. Diseño de la solución

3.1 Problemática que debe resolverse

El uso de las redes SDN está extendiéndose. Las redes SDN han estado orientadas en un inicio a las redes cableadas, pero la realidad es que si las redes SDN deben sustituir a las redes tradicionales su integración con todas las formas de acceso a la red debe ser total.
Actualmente, a nivel empresarial cuando es necesario desplegar una nueva característica en la red, por ejemplo un firewall, el tiempo para llevar a cabo el despliegue del equipo y la configuración del mismo es demasiado elevado. Las redes SDN y las NFV permiten que el despliegue de este tipo de elementos en la red sea cuestión de pocos segundos.
Uno de estos elementos son los equipos controladores WiFi. Como se ha explicado anteriormente, las redes WiFi empresariales se basan en un equipo controlador y una serie de APs que se asocian con el mismo. Por lo general, las empresas tienen varias sedes que están interconectadas por una red MPLS. El servicio de acceso a la red por WiFi presenta múltiples beneficios y es por ello que las empresas dotan a todas sus sedes de acceso tanto cableado como inalámbrico. Al tener que gestionar APs en varias sedes surgen problemas, sobre todo grandes latencias entre APs y controlador. Las grandes latencias hacen que la red no funcione al rendimiento esperado y consigue que la experiencia del usuario no sea satisfactoria.
A parte, las empresas que tienen un caracter industrial se ven en la necesidad de cablear en red infinitud de dispositivos IoT, por ejemplo PLCs, controles numéricos, sensores, grúas, etc. Estos dispositivos en la mayoría de ocasiones tienen la capacidad de conectarse a red por WiFi, pero habitualmente no se realiza ese tipo de conexionado por pensar que la red WiFi no ofrece una calidad adecuada debido a la experiencia de usuario.
La solución a este problema es el despliegue de un controlador WiFi por cada una de las sedes de la empresa, lo cual resulta ineficiente en varios sentidos. Es necesario realizar la compra de un controlador WiFi para cada una de las sedes, configurarlo y mantenerlo de manera individual, lo que supone unos grandes costes para las empresas.
El objetivo del presente trabajo es diseñar una arquitectura de red inalámbrica empresarial, en la que el equipo controlador WiFi esté virtualizado como una función de red. Gracias a ello, el despliegue de un equipo controlador se podrá hacer mediante el encendido de una máquina virtual y la configuración del mismo se hará de manera genérica a través de una aplicación. La gestión de los recursos inalámbricos de la red y el aprovisionamiento de APs sigue siendo responsabilidad del controlador WiFi, pero el control de los flujos inalámbricos es responsabilidad del controlador SDN. Todos los aspectos relativos al control de los flujos inalámbricos se llevará a cabo mediante aplicaciones que indicarán al controlador qué flujos instalar en los switches.
Con esto se aumenta el rendimiento de la red inalámbrica y a parte, se permite que los costes asociados a la adquisición, despliegue y configuración de los controladores se vean minimizados.

3.2 Objetivos y requisitos

Como se ha expuesto en el apartado anterior el objetivo es virtualizar el equipo controlador WiFi como una función virtual de red. Una vez hecho esto es necesario programar aplicaciones que interactúen con la interfaz norte del controlador de la red. Estas aplicaciones tendrán la función de definir los flujos que el controlador debe instalar en los switches de la red.

3.3 Arquitectura propuesta

3.3.1 Controlador SDN

Después de la comparación entre controladores hecha en el apartado 2.1.2 se escoge Ryu como controlador para este proyecto. Es un controlador que tiene una API bien definida, permite el uso de REST en la interfaz norte, soporta varias versiones de OpenFlow y se integra con Open vSwitch por lo que cumple con los requisitos para ser utilizado en este proyecto. Además, Ryu se ejecuta como una aplicación de Python, lo que hace que sea muy ligero.

Ryu es un entorno de trabajo que permite implementar redes SDN mediante software. Algunas de las características que ofrece son:

- Capacidad de escuchar eventos asíncronos
- Capacidad de análisis de paquetes
- Capacidad de interactuar mediante OpenFlow
- Agilidad para crear una infraestructura SDN
- Flexibilidad en la interfaz Norte

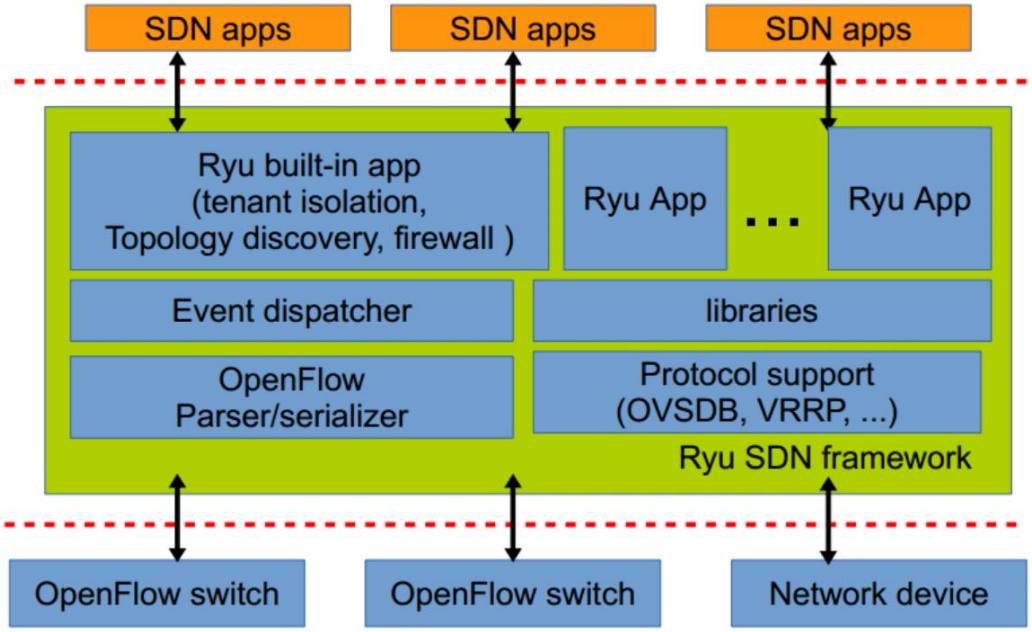


Figura 13 – Arquitectura de Ryu
Fuente:
https://www.researchgate.net/profile/Indrarini_Irawati/publication/282754539/figure/fig2/AS:651667431235586@1532381031504/Ryu-SDN-framework-6.png

La utilización de Ryu permite gestionar eventos de red y reaccionar a ellos instalando flujos si es necesario, esto lo hace mediante aplicaciones.
Ryu cuenta a su vez con varias librerías con protocolos como Netconf, vrrp o Netflow.

3.3.2 Controlador WiFi

Para virtualizar el equipo controlador WiFi se ha elegido el software Mikrotik [24]. Mikrotik es una empresa de la República de Latvia que se encarga de desarrollar un sistema operativo para equipos de red. Ese sistema operativo puede obtenerse dentro de un appliance físico (de bajo coste) o virtualizarlo en cualquier plataforma. Es un software muy potente que está basado en el Kernel de Linux, ofrece varias características interesantes: puede dar servicio a entornos inalámbricos, tiene opciones de routing dinámicas, contiene firewall integrado, permite levantar túnels IPsec, etc. En este caso concreto la característica más importante que ofrece el software es la capacidad de actuar como equipo controlador WiFi. Cuenta con la capacidad de utilizar el protocolo CAPWAP para aprovisionar a los APs y tiene dos modos de funcionamiento (forwarding y local forwarding). La diferencia principal entre ambos modos es que con el modo forwarding todos los flujos inalámbricos van hasta el controlador, mientras que con el modo local forwarding no es necesario ya que el tráfico lo gestionan los switches sin necesidad de pasar por el controlador.
Los equipos Mikrotik cuentan con una API para configurar los equipos de manera dinámica y además soportan el protocolo OpenFlow, por lo que pueden integrarse con cualquier controlador SDN.

3.3.3 Infraestructura

En cuanto a la infraestructura de red son necesarios dos elementos, switches con soporte para el protocolo OpenFlow y APs WiFi.
En el caso de los switches hay multitud de opciones, pero en este caso se escoge Open vSwitch [25].
Open vSwitch es un software de código abierto con licencia bajo el proyecto de Apache 2. El objetivo del software es el de implementar un switch que soporte interfaces de gestión estándar y permita gestionar las funciones de re-envío a través de programación.

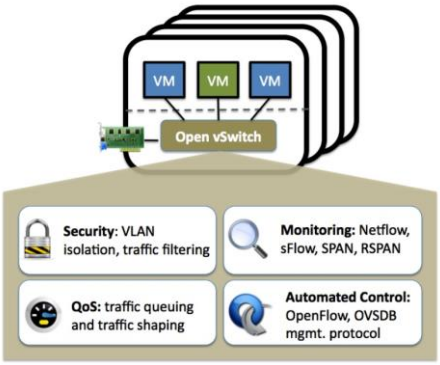


Figura 14 – Open vSwitch
Fuente: https://docs.openvswitch.org/en/latest/_images/overview.png

Open vSwitch tiene los siguientes componentes:

- ovs-vswitchd: Demonio que implementa el switch y la conmutación basada en flujos.
- ovsdb-server: Servicio de bases de datos para almacenamiento de la configuración.
- ovs-dpctl: Herramienta para configurar el módulo del switch.
- ovs-vsctl: Herramienta para configurar el módulo ovs-vswitchd.
- ovs-appctl: Utilidad para enviar comandos a los demonios de Open vSwitch.
- ovs-controller: Un controlador OpenFlow
- ovs-ofctl: Utilidad para controlar los contenidos de la tabla de flujos si el switch está funcionando como switch OpenFlow.

Principalmente el ser una herramienta de código abierto, virtualizable y con soporte para el protocolo OpenFlow hacen de este switch idóneo para el proyecto.

En el caso de APs WiFi se plantea el uso de equipos Mikrotik. Tal y como se comentaba antes el software Mikrotik es muy potente y otra de las opciones que ofrece es la de convertir a cualquier equipo en un AP (por supuesto esto viene condicionado a que el equipo físico en cuestión tenga una tarjeta de red inalámbrica que pueda soportarlo). Estos equipos son interesantes para ser utilizados como APs por dos aspectos:

- Se permite la asociación con el controlador WiFi para aprovisionamiento de configuración.
- Se soporta el uso del protocolo OpenFlow, lo que permite controlar al AP desde el controlador como un switch más.

3.3.4 Virtualización de funciones de red

A parte de la virtualización del equipo controlador WiFi es necesario introducir una función de red virtualizada más. En toda red WiFi se tiene que proporcionar un mecanismo de autenticación de clientes, y por lo general en las redes empresariales esta autenticación la realiza un servidor Radius que se comunica con el equipo controlador. Se utiliza para ello el protocolo IEEE 802.1ax.
Es por ello que es necesario incluir un servidor Radius virtualizado, el cuál tendrá la función principal de tener la base de datos de clientes que tengan acceso a la red. Cada vez que un cliente quiera acceder a la red mandará la solicitud al equipo controlador y será este el que se comunique con el servidor Radius para comprobar si el cliente está o no en la base de datos.

3.3.5 Virtualización de infraestructura

Para llevar a cabo la virtualización de la infraestructura, en este caso servidor Radius y controladora WiFi se ha decidido utilizar KVM. Como se ha explicado anteriormente, KVM es la plataforma de virtualización de Linux. Permite el despliegue de máquinas virtuales y gestionarlas de manera remota.

3.3.6 Aplicaciones

Es necesario la utilización de varias aplicaciones para conseguir la automatización de la red inalámbrica.

Aplicaciones para configuración de equipos inalámbricos:

- Aplicación para configuración de equipo controlador. La finalidad de esta aplicación es la convertir un equipo Mikrotik en controlador WiFi y la de desplegarle la configuración que deben tener los APs. Esta configuración indicará a los APs que SSIDs radiar, en qué frecuencia, etc. Se utilizará la API del equipo.
- Aplicación para la configuración de AP. La finalidad de esta aplicación es la de convertir cualquier equipo Mikrotik en AP e indicarle con qué controladora debe asociarse. Se utilizará la API del equipo.

Aplicaciones para control de flujos:

- Aplicación para la gestión de flujos. La finalidad de esta aplicación es la de controlar la tabla de los switches OpenFlow. Se utilizará la API de la interfaz norte del controlador.
- Aplicación de firewall. La finalidad de esta aplicación es la de aplicar un firewall virtual a los equipos que se deseen. Se utilizará la API de la interfaz norte del controlador.

4. Laboratorio de pruebas

4.1 Arquitectura de pruebas

El esquema de red virtual que va a utilizarse para las pruebas es el siguiente:

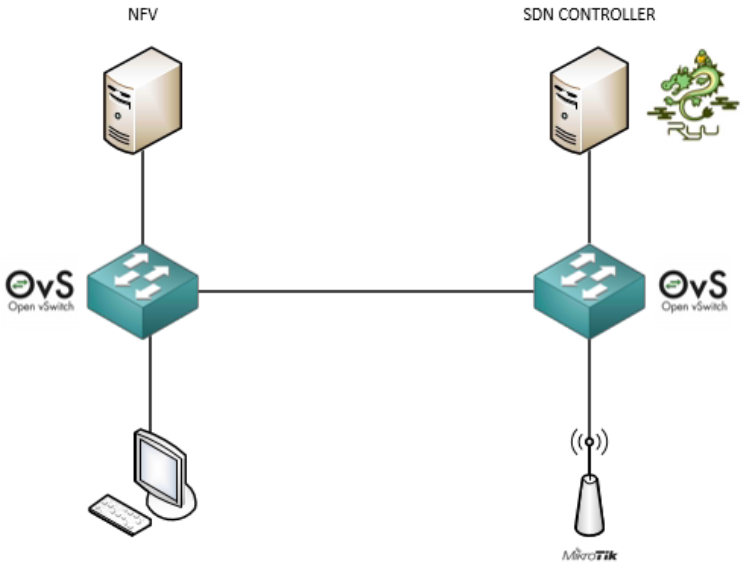


Figura 15 – Arquitectura lógica para pruebas
Fuente: Propia

Como se puede apreciar, el esquema lógico se compone de:

- 1 servidor 'SDN CONTROLLER'. Este servidor aloja tanto el controlador SDN (Ryu) como las aplicaciones que se utilizan para gestionar el controlador SDN a través de la interfaz norte. A parte, este es el equipo que se utiliza para realizar la configuración de los elementos de la red.
- 1 servidor 'NFV'. Este servidor tiene la función de soportar la virtualización de funciones de red por lo que tiene instalado el software KVM de virtualización de Linux. En este servidor se virtualizan dos funciones de red, el servidor Radius y la controladora WiFi.
- 2 switches Open vSwitch. Estos dos switches hacen de switches de core para permitir la comunicación entre todos los elementos de la red. Se configuran para integrarse con el controlador para que sea este mediante aplicaciones y OpenFlow el que los gestione.
- 1 PC virtual que simule un cliente estando conectado a través de cable de red.
- 1 AP que permita la conectividad de dispositivos de manera inalámbrica a la red.

Esta maqueta pretende simular un escenario corporativo que permite un acceso híbrido a la red, tanto por cable como por WiFi. Más concretamente se pretende simular un escenario corporativo industrial que cuenta con gran cantidad de dispositivos IoT, los cuales es necesario conectar por WiFi.

En escenarios corporativos con empresas que tienen gran cantidad de sedes se hace conveniente el uso de SDN para la gestión de la red. SDN permite disminuir los costes operacionales y de inversión. En este caso concreto, en escenarios con gran cantidad de switches se hace conveniente y deseable el no tener que configurar todos y cada uno de ellos individualmente. En este trabajo se presentan dos aplicaciones para la configuración de manera remota y sencilla de controladoras WiFi y de APs.
Adicionalmente, el despliegue de la controladora WiFi presenta estas mejoras con respecto a un despliegue de red convencional:

- Tiempo de despliegue bajo, para instalar una nueva controladora WiFi solo es necesario el despliegue de una máquina virtual en un servidor.
- Posibilidad de desplegar nuevas controladoras y apagarlas posteriormente ante picos de demanda puntuales, permitiendo dividir los APs.
- No es necesario personal para hacer trabajo físico ya que no es necesario realizar conexionado de cables ni enrackado de equipos.

4.1.1 Despliegue de la arquitectura

La maqueta de pruebas se despliega mediante la aplicación GNS3. GNS3 es una herramienta de simulación de redes que se integra con diferentes herramientas de virtualización, como VirtualBox o VMware. Esta herramienta permite introducir en una maqueta de red aislada máquinas virtuales que están creadas en cualquiera de las herramientas de virtualización soportadas.
Además, la herramienta da la posibilidad de conectar la maqueta de red al exterior usando el equipo host como puente. Otra de las utilidades que ofrece GNS3 es la de hacer capturas de Wireshark selectivas en los enlaces que se desee.

En este caso la maqueta desplegada en GNS3 es la siguiente:

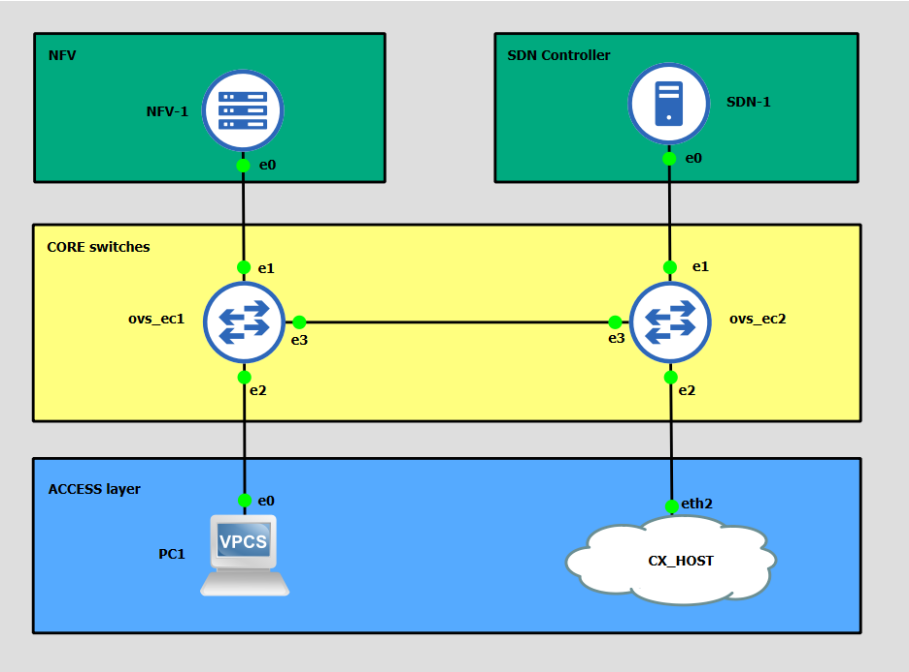


Figura 16 – Maqueta GNS3
Fuente: Propia

Físicamente la maqueta es la siguiente:

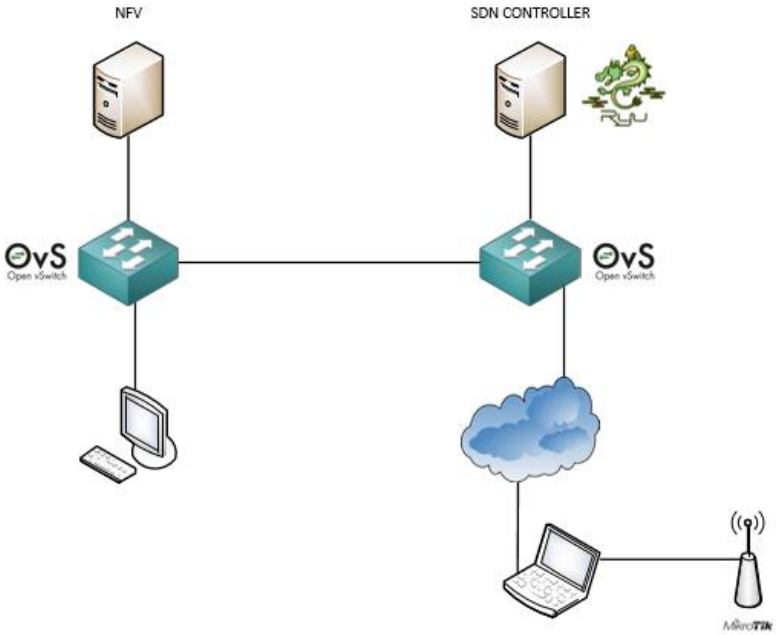
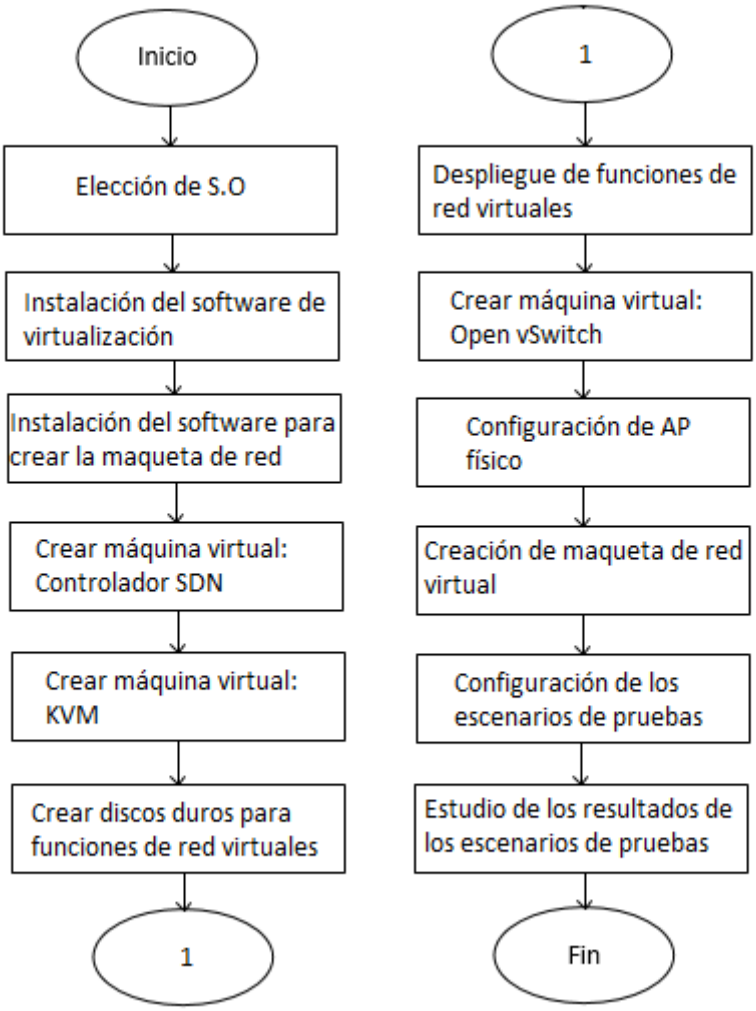


Figura 17 – Arquitectura física
Fuente: Propia

La maqueta física es igual que la lógica con la excepción de que para realizar la conectividad de la maqueta virtual con el AP físico se usa el equipo que aloja la maqueta como puente. Se configura el adaptador de red del equipo de manera manual para que forme parte de la maqueta como si fuese un equipo más. El software GNS3 permite mediante una nube conectar la maqueta con un adaptador de red (virtual o físico del equipo host). En este caso se utiliza un adaptador físico para poder conectar el AP y que también forme parte de la red.

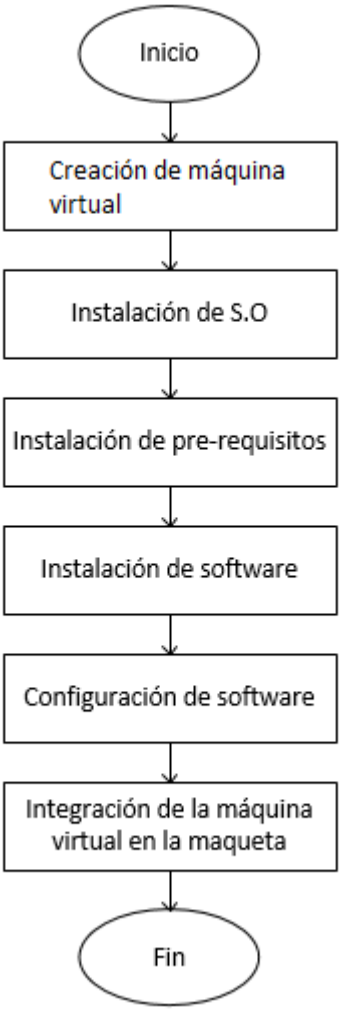
Para desplegar la maqueta de red y realización de pruebas sobre ella se sigue el siguiente diagrama de flujo



4.1.2 Configuración de los elementos de red de la arquitectura

Todas las configuraciones de red de base de los equipos se añaden en el Anexo 1.

Para realizar la instalación de las máquinas virtuales se sigue el siguiente diagrama de flujo que es común para todas ellas.



Controlador SDN (Ryu)

En el caso del controlador es una aplicación de Python, por lo que es necesario realizar la instalación del software. Una vez instalado el software el controlador Ryu se instala como una aplicación más de Python. Con la aplicación instalada se instalan también dos aplicaciones necesarias para realizar configuraciones remotas en equipos de red.

La instalación de todos los elementos se hace de la siguiente manera:

Instalación de los pre-requisitos:

```
# apt-get install python3
# apt-get install python3-pip
# apt-get install gcc python3-dev libffi-dev libssl-dev libxml2-dev libxslt1-dev zlib1g-dev
# apt-get install curl
```

Instalación del controlador:

```
# pip3 install ryu
```

Instalación de los complementos necesarios:

```
# pip3 install paramiko
# pip3 install librouteros
```

Paramiko es un componente de Python que permite la interacción con dispositivos de red por medio de SSH/Telnet y librouteros es un componente que permite la interacción con la API de Mikrotik.

Para arrancar el controlador simplemente hay que ejecutar una o varias aplicaciones:

```
# ryu run app1.py app2.py
```

NFV

Para desplegar las máquinas virtuales que ejecutarán las funciones de red virtuales es necesario tener un software que actúe como hipervisor.
En este caso es necesaria la instalación de KVM para permitir la virtualización de los elementos necesarios.

```
# apt-get install qemu-kvm libvirt-bin bridge-utils virtinst virt-manager
```

Con la máquina instalada se define la red para las máquinas virtuales:

```
# cat host-bridge.xml
<network>
  <name>host-bridge</name>
  <forward mode="bridge"/>
  <bridge name="br0"/>
</network>
# virsh net-define host-bridge.xml
# virsh net-start host-bridge
# virsh net-autostart host-bridge
```

Una vez hecho esto, ya se está en disposición de desplegar máquinas virtuales. Es posible realizarlo tanto por línea de comandos como de manera gráfica. En este caso la manera elegida para desplegar las máquinas virtuales es mediante el uso de ficheros *.qcow2. Los ficheros *.qcow2 son discos duros virtuales que pueden referenciarse a la hora de crear una máquina virtual. Esto posibilita el tener ya pre-configurado el disco duro virtual de una máquina, de manera que si quiere desplegarse una máquina virtual sólo sería necesario copiarlo en donde se desse desplegar la máquina y referenciarlo.

En este caso se despliegan dos máquinas virtuales

```
# virt-install --name freeradius --memory 512 --vcpus=1 --disk /home/sergio/Documentos/freeradius.qcow2,bus=ide --import --os-type=generic --os-variant=generic --network bridge=br0
```

```
# virt-install --name mikrotik --memory 128 --vcpus=1 --disk /home/sergio/Documentos/mikrotik.qcow2,bus=ide --import --os-type=generic --os-variant=generic --network bridge=br0
```

Una vez desplegadas las máquinas virtuales se configuran para que autoarranquen al iniciarse el sistema operativo

```
# virsh autostart freeradius
# virsh autostart mikrotik
```

Freeradius

Una de las funciones virtuales de red que son necesarias es un servidor radius. Este servidor radius tiene la función de autenticar a los clientes WiFi.
En este caso se ha elegido el servidor freeradius, ya que es opensource.

Instalación del servidor Freeradius:

```
# apt-get install freeradius
```

El servidor Freeradius va a utilizarse para proveer autenticación 802.1ax dentro de la red inalámbrica. Para esto se utiliza el módulo EAP que es necesario habilitar.

```
# ln -s /etc/freeradius/3.0/mods-available/eap /etc/freeradius/3.0/mods-enabled/eap
```

Dentro del fichero eap se definen los diferentes modos de acceso

```
# vi /etc/freeradius/3.0/mods-available/eap
eap {
default_eap_type = md5
timer_expire = 60
ignore_unknown_eap_types = no
cisco_accounting_username_bug = no
max_sessions = 4096
md5 {
}
...
}
```

Dentro del fichero usuarios se definen los usuarios que tienen acceso a la red

```
# vi /etc/freeradius/3.0/users
Usuario1 Cleartext-Password:="usuario1"
Usuario2 Cleartext-Password:="usuario2"
```

Esta es una manera de realizar la definición de usuarios, otra manera es la de utilizar una base de datos o un servidor de autenticación externo como por ejemplo el Directorio Activo de Microsoft. Para este entorno de pruebas se considera suficiente definir de esta manera los usuarios.

Por último, es necesario declarar el equipo que va a lanzar las autenticaciones al servidor Radius.

```
# vi /etc/freeradius/3.0/clients.conf
client 192.168.58.250 {
        secret = wfcorp_pass
        shortname = capsman
}
```

Controladora WiFi virtual

La otra función de red virtual que se utiliza es una controladora de red virtual. Son necesarias varias configuraciones en el equipo Mikrotik que actua como controladora WiFi.

En primer lugar un servidor DHCP:

```
/ip pool add name=pool1 ranges=192.168.58.100-192.168.58.200
/ip dhcp-server network add address=192.168.58.0/24
/ip dhcp-server add address-pool=pool1 disabled=no interface=lan name=server1
```

En segundo lugar es necesario apuntar hacia el servidor radius e indicar la clave definida anteriormente:

```
/radius add address=192.168.58.251 secret=wfcorp_pass service=wireless
```

Por último hay que habilitar el equipo como controladora WiFi en varios pasos.

Se define el tipo de seguridad de la red WiFi

```
/caps-man security add authentication-types=wpa2-eap eap-methods=passthrough encryption=aes-ccm group-encryption=aes-ccm name=security1 tls-certificate=none tls-mode=no-certificates
```

Se crea el fichero de configuración que va a desplegarse a los APs de manera automática. Se indica en este paso también que los APs no deben mandar el tráfico hasta la controladora, sino que debe estar gestionado por los switches más cercanos al AP en cuestión.

```
/caps-man configuration add country=spain datapath.bridge=lan datapath.client-to-client-forwarding=yes datapath.local-forwarding=yes name=cfg1 security=security1 ssid=wf_corp
/caps-man provisioning add action=create-dynamic-enabled master-configuration=cfg1
```

Por último se habilita el equipo como controladora WiFi

```
/caps-man manager interface add disabled=no interface=lan
/caps-man manager set enabled=yes
```

Switches Open vSwitch

El software de Open vSwitch es muy versátil y puede instalarse sobre cualquier máquina Linux. En este caso se ha decidido usar una máquina virtual independiente por cada uno de los switches. Estas dos máquinas van a utilizarse como los switches corporativos.

Para instalar Open vSwitch sobre una máquina Linux se hace de la siguiente manera:

```
# apt-get install openvswitch-switch openvswitch-common
```

Una vez instalado, se crea un switch virtual al que se asignan todos los puertos de la máquina virtual en una vlan concreta. Para dicha VLAN se crea una

interfaz de nivel 3 en el equipo. Esta interfaz se utiliza para alcanzar el controlador SDN. Por último, se enlaza el switch virtual con el controlador SDN.

```
# ovs-vsctl add-br sw0
# ovs-vsctl add-port sw0 enp0s3
# ovs-vsctl add-port sw0 enp0s8
# ovs-vsctl add-port sw0 enp0s9
# ovs-vsctl add-port sw0 enp0s10
# ovs-vsctl set port enp0s3 tag=58 vlan_mode=native-untagged
# ovs-vsctl set port enp0s8 tag=58 vlan_mode=native-untagged
# ovs-vsctl set port enp0s9 tag=58 vlan_mode=native-untagged
# ovs-vsctl set port enp0s10 tag=58 vlan_mode=native-untagged
# ovs-vsctl add-port sw0 vlan58 tag=58 -- set interface vlan58 type=internal
# ovs-vsctl set-controller sw0 tcp:192.168.58.253:6633
```

Debido a que las máquinas virtuales creadas en VirtualBox sólo pueden tener 4 adaptadores de red los switches son de 4 puertos cada uno. Se configuran dos exactamente igual con dos IPs diferentes.

PC1

El PC1 es un terminal cmd emulado por GNS3 que se va a utilizar simplemente para comprobar conectividad de red. Se configura para que solicite una IP por DHCP. Esta IP vendrá dada por la controladora WiFi.

```
PC1> dhcp
```

HOST

Para realizar la conexión de la maqueta con el AP físico es necesario configurar el PC Host como puente. Para ello se configura el adaptador de red del equipo dentro de la misma red que se ha utilizado en la maqueta y se conecta mediante una nube de GNS3.

AP

Como AP se va a utilizar un equipo Mikrotik, al igual que la controladora. La diferencia entre el equipo que hace de controladora y el AP es que el AP necesita una tarjeta de red inalámbrica mientras que la controladora no. Esta tarjeta se utiliza para radiar la red WiFi.
Para configurar el equipo Mikrotik como AP se siguen estos pasos:

Se habilita el equipo como dhcp relay para lanzar las consultas DHCP a la controladora WiFi:

```
/ip dhcp-relay add dhcp-server=192.168.58.250 disabled=no interface=bridgeTFM local-address=192.168.58.229 name=relay1
```

Se configura el equipo como AP indicando con qué controladora debe asociarse:

```
/interface wireless cap set bridge=bridgeTFM caps-man-addresses=192.168.58.250 discovery-interfaces=bridgeTFM enabled=yes interfaces=wlan1
```

Un resumen del direccionamiento asignado es el siguiente:

HOST	IP	MAC
SDN Controller	192.168.58.253	08:00:27:E1:ED:3F
NFV	192.168.58.252	08:00:27:3F:9D:55
Radius	192.168.58.251	52:54:00:56:99:9F
WLC	192.168.58.250	52:54:00:50:AA:81 48:8F:5A:9A:6F:3F
OVS-EC1	192.168.58.240	08:00:27:00:55:DD
OVS-EC2	192.168.58.239	08:00:27:0B:62:87
AP	192.168.58.229	48:8F:5A:9A:6F:3A 48:8F:5A:9A:6F:3F
PC1	DHCP	00:50:79:66:68:00
RASPBERRY	DHCP	B8:27:EB:A9:31:F1
MÓVIL	DHCP	0A:42:17:87:07:61
GNS3 VM	192.168.58.2	08:00:27:15:79:CF
HOST	192.168.58.1	B4:B6:86:26:C1:6F

El direccionamiento asignado por DHCP es el rango 192.168.58.100 – 192.168.58.200

4.1.3 Aplicaciones de red

Existen dos casuísticas en la red, una en la que los switches OpenFlow se comportan como tal y otra en la que los switches OpenFlow se comportan como firewalls.

Casuística switch

En este caso se utilizan dos aplicaciones:

- simple_switch_rest_13.py : Esta aplicación se ejecuta junto con el controlador SDN y hace que los equipos OpenFlow se comporten como switches normales. Se hace uso de la versión 1.3 de OpenFlow. Adicionalmente, la aplicación expone una API REST a través de la cual interactuar con el controlador.

Método	URL	Acción
GET	http://{CONTROLADOR}:8080/ simpleswitch/mactable/{SWITCH ID}	Obtener tabla de MACs de un switch
PUT	http://{CONTROLADOR}:8080/ simpleswitch/mactable/{SWITCH ID}	Introducir una entrada en la tabla de MACs de un switch. Los datos van en el cuerpo del mensaje en formato JSON.

- switch_management.py : Esta aplicación se ejecuta de manera independiente al controlador y sirve para actuar de una manera intuitiva

sobre la API REST de la primera aplicación. Ofrece la posibilidad de consultar la tabla de MACs, formatearla de manera legible y también la posibilidad de introducir entradas manuales en las tablas MAC de los switches.

Casuística firewall:

En este caso se utilizan dos aplicaciones:

- rest_firewall.py : Esta aplicación se ejecuta junto con el controlador SDN y hace que los equipos OpenFlow se comporten como firewalls. Se hace uso de la versión 1.3 de OpenFlow. Adicionalmente, la aplicación expone una API REST a través de la cual interactuar con el controlador.

Método	URL	Acción
GET	http://{CONTROLADOR}:8080/ /firewall/module/status	Obtener el estado del firewall en los switches de la topología
PUT	http://{CONTROLADOR}:8080/ /firewall/module/status/enable/{SWITCH ID}	Habilitar la característica de firewall en un switch
PUT	http://{CONTROLADOR}:8080 /firewall/module/status/disable/{SWITCH ID}	Deshabilitar la característica de firewall en un switch
GET	http://{CONTROLADOR}:8080 /firewall/rules/{SWITCH ID}	Obtener las reglas habilitadas en un switch
POST	http://{CONTROLADOR}:8080 /firewall/rules/{SWITCH ID}	Introducir nuevas reglas en un switch. Los datos van en el cuerpo del mensaje en formato JSON.
DELETE	http://{CONTROLADOR}:8080 /firewall/rules/{SWITCH ID}	Borrar una regla de un switch. Los datos van en el cuerpo del mensaje en formato JSON.

- firewall_management.py : Esta aplicación se ejecuta de manera independiente al controlador y sirve para actuar de una manera intuitiva sobre la API REST de la primera aplicación. Ofrece la posibilidad de habilitar/deshabilitar el firewall en los switches, consultar las reglas insertadas en los switches, introducir reglas y eliminarlas.

4.2 Pruebas sobre arquitectura de red

Para realizar pruebas se plantean tres casos de uso que pueden darse en cualquier escenario corporativo industrial cuando se conectan dispositivos a la red.

- Caso 1: Conseguir comunicación entre dos dispositivos IoT conectados mediante WiFi.
- Caso 2: Conseguir comunicación entre un dispositivo IoT conectado por WiFi y un dispositivo conectado por cable de red.
- Caso 3: Conseguir filtrar la comunicación entre un dispositivo IoT conectado por WiFi y un dispositivo conectado por cable de red haciendo que los switches de la maqueta se comporten como Firewalls.

4.2.1 Caso de uso 1

Para este caso de uso se conectan dos equipos por WiFi para comprobar la conectividad entre los mismos. Los objetivos de este caso son:

- Comprobar que la controladora WiFi ha desplegado correctamente la configuración en el AP.
- Comprobar que es posible asociar dispositivos a la red WiFi.
- Comprobar que hay comunicación entre los dos dispositivos conectados.
- Comprobar que el tráfico entre los dos dispositivos no viaja hasta la controladora

Para comprobar los objetivos marcados en el caso 1 es necesario lo siguiente:

- Comprobar que el AP WiFi radia la red inalámbrica configurada en la controladora.
- Comprobar que cualquier dispositivo es capaz de autenticarse en la red con un usuario definido en el servidor Radius.
- Comprobar cualquier dispositivo asociado a la red WiFi obtiene una IP del servidor DHCP.
- Comprobar que hay conectividad entre los elementos de red mediante el uso del protocolo de red ICMP.
- Comprobar que no hay flujos instalados en los switches relativos al tráfico generado entre dispositivos.

La arquitectura de red para este caso es la siguiente:

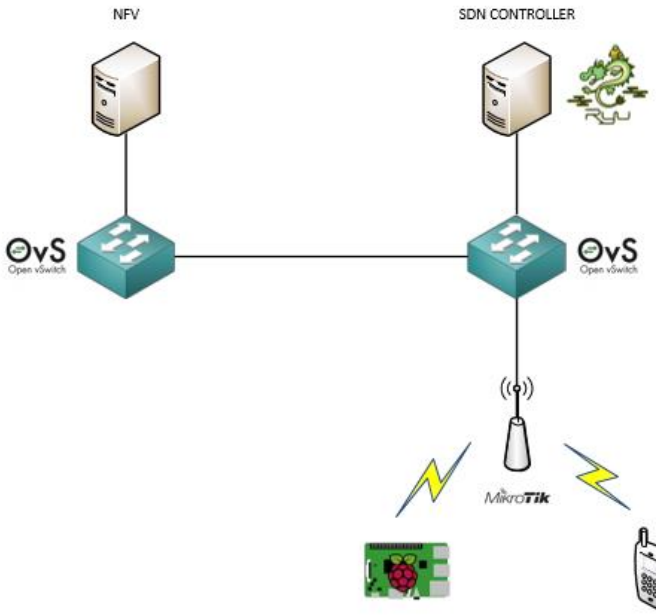


Figura 18 – Arquitectura para caso 1
Fuente: Propia

En este caso la arquitectura de red se compone del servidor con las funciones de red virtuales, el controlador sdn, dos switches, un AP y como dispositivos IoT se usan una raspberry pi y un móvil.

En primer lugar se asocian tanto la Raspberry PI como el teléfono móvil por WiFi al AP. En la controladora WiFi se pueden ver las concesiones DHCP por lo que se comprueba que los equipos asociados por WiFi obtienen una dirección IP.

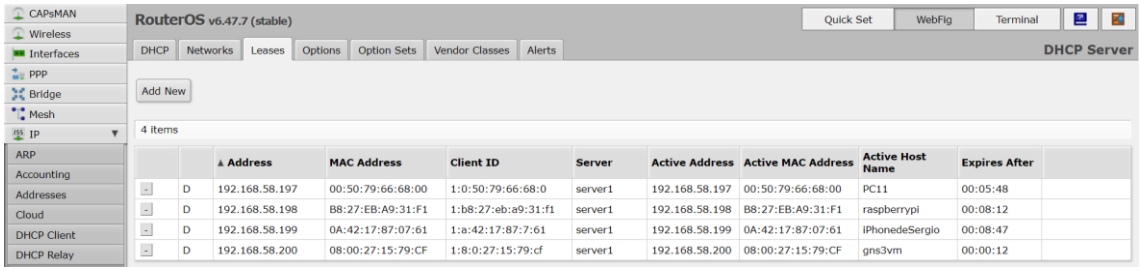


Figura 19 – DHCP Caso 1
Fuente: Propia

En el AP se puede ver que el equipo está gestionado por una controladora WiFi y que está funcionando de manera local para la gestión del tráfico. También se aprecia el SSID WiFi que se radia el AP.

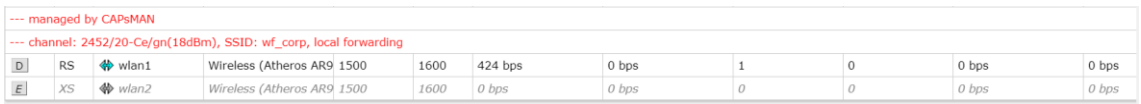


Figura 20 – AP Caso 1
Fuente: Propia

En las siguientes capturas se puede apreciar la IP configurada en cada uno de los dispositivos.

Red para la Raspberry Pi:

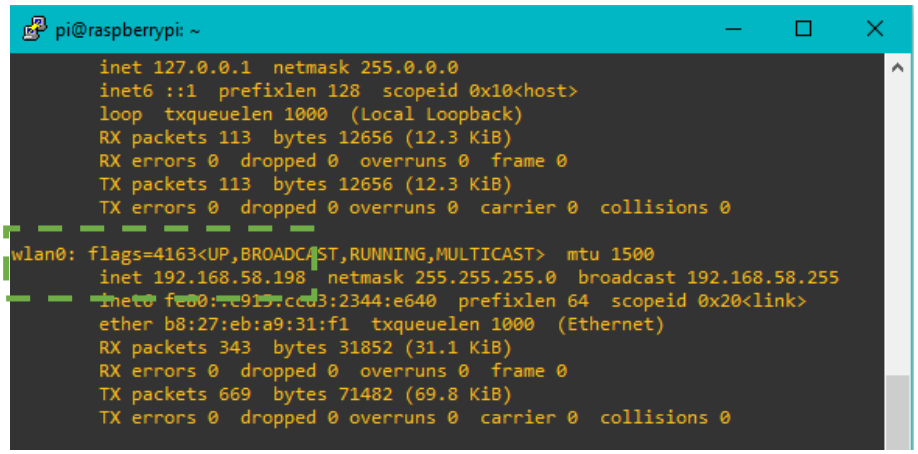


Figura 21 – Raspberry Caso 1
Fuente: Propia

Red para el teléfono móvil, se puede ver la IP y el SSID WiFi:

Figura 22 – Movil Caso 1
Fuente: Propia

Viendo que los dos equipos están asociados a la red WiFi que radia el AP se comprueba que son capaces de autenticarse en la red.

Se ejecuta el controlador con la aplicación switch simple que trae por defecto

```
root@SDN:/usr/local/lib/python3.6/dist-packages/ryu/app# ryu run simple_switch.py
loading app simple_switch.py
loading app ryu.controller.ofp_handler
instantiating app simple_switch.py of SimpleSwitch
instantiating app ryu.controller.ofp_handler of OFPHandler
```

Figura 23 – Controlador Caso 1
Fuente: Propia

Se lanza un ping desde el terminal de la Raspberry al móvil.

```
root@raspberrypi:/home/pi# ping 192.168.58.199
PING 192.168.58.199 (192.168.58.199) 56(84) bytes of data.
64 bytes from 192.168.58.199: icmp_seq=1 ttl=64 time=21.9 ms
64 bytes from 192.168.58.199: icmp_seq=2 ttl=64 time=370 ms
64 bytes from 192.168.58.199: icmp_seq=3 ttl=64 time=428 ms
64 bytes from 192.168.58.199: icmp_seq=4 ttl=64 time=520 ms
64 bytes from 192.168.58.199: icmp_seq=5 ttl=64 time=134 ms
64 bytes from 192.168.58.199: icmp_seq=6 ttl=64 time=54.5 ms
64 bytes from 192.168.58.199: icmp_seq=7 ttl=64 time=690 ms
64 bytes from 192.168.58.199: icmp_seq=8 ttl=64 time=713 ms
64 bytes from 192.168.58.199: icmp_seq=9 ttl=64 time=433 ms
64 bytes from 192.168.58.199: icmp_seq=10 ttl=64 time=251 ms
64 bytes from 192.168.58.199: icmp_seq=11 ttl=64 time=66.0 ms
^C
--- 192.168.58.199 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 20ms
rtt min/avg/max/mdev = 21.869/334.648/713.446/237.214 ms
```

Figura 24 – Prueba conectividad Caso 1
Fuente: Propia

Se confirma que hay conectividad de red y se comprueban los flujos instalados en los dos switches para verificar que no hay flujos asociados a esta comunicación, tal y como debería ser.

```
root@ovs_ec1:/home/sergio# ovs-ofctl dump-flows sw0 | grep b8:
root@ovs_ec1:/home/sergio#
```

Figura 25 – Flujos SW 1 Caso 1
Fuente: Propia

```
root@ovs_ec2:/home/sergio# ovs-ofctl dump-flows sw0 | grep b8
 cookie=0x0, duration=9.418s, table=0, n_packets=11, n_bytes=706, idle_age=1, in_port=3,dl_src=b4:b6:86:26:c1:6f,dl_dst=b8:27:eb:a9:31:f1 actions=output:3
 cookie=0x0, duration=9.341s, table=0, n_packets=12, n_bytes=1706, idle_age=0, in_port=3,dl_src=b8:27:eb:a9:31:f1,dl_dst=b4:b6:86:26:c1:6f actions=output:3
root@ovs_ec2:/home/sergio#
```

Figura 26 – Flujos SW2 Caso 1
Fuente: Propia

Como se puede apreciar el switch 1 no tiene flujos asociados para la raspberry y el switch 2 sí que tiene uno, pero para la comunicación entre la raspberry y el equipo host. No hay flujos de comunicación entre la raspberry y el móvil.
Con esto se comprueban todos los puntos marcados como objetivos para el caso 1.

4.2.2 Caso de uso 2

Para este caso de uso se conecta un equipo por WiFi y otro equipo por cable. El esquema de acciones a realizar en este caso es:

- Comprobar que hay comunicación entre los dos dispositivos conectados.
- Verificar los flujos OpenFlow que se instalan en los switches involucrados.
- Estudiar los mensajes OpenFlow que se envían entre el controlador y los switches.

Para comprobar los objetivos definidos en el caso de uso 2 es necesario:

- Verificar mediante protocolo ICMP que hay conectividad de red entre los dos dispositivos.
- Verificar mediante línea de comandos los flujos instalados en cada switch relativos a la comunicación entre dispositivos.
- Realizar y estudiar capturas de Wireshark en los enlaces relevantes de la maqueta para ver los paquetes OpenFlow desde y hacia el controlador.

La arquitectura de red para este caso es la siguiente:

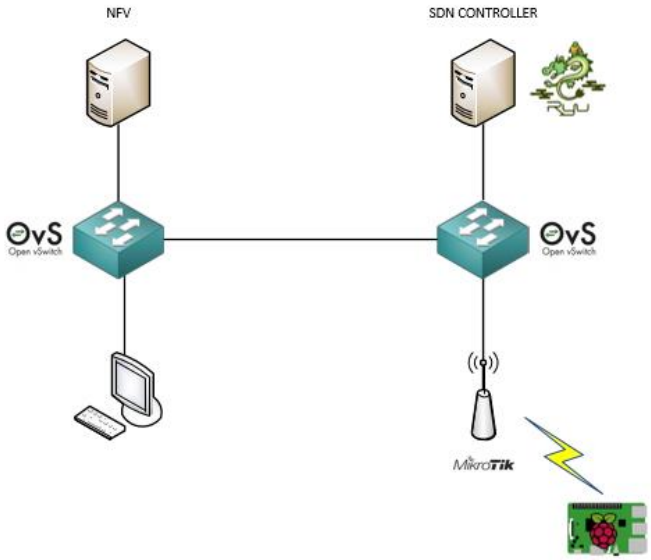


Figura 27 – Arquitectura para pruebas caso 2
Fuente: Propia

En este caso se usa la arquitectura propuesta en el punto 4.1 teniendo conectado por WiFi un dispositivo IoT (raspberry pi).

Se configura la red para ambos dispositivos involucrados:

Red para el PC:

```
PC1> show ip

NAME        : PC1[1]
IP/MASK     : 192.168.58.199/24
GATEWAY     : 0.0.0.0
DNS         :
DHCP SERVER : 192.168.58.250
DHCP LEASE  : 594, 600/300/525
MAC         : 00:50:79:66:68:00
LPORT       : 10027
RHOST:PORT  : 127.0.0.1:10028
MTU:        : 1500
```

Figura 28 – PC1 Caso 2

Fuente: Propia

Red para la Raspberry Pi:

```
pi@raspberrypi: ~

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.58.197  netmask 255.255.255.0  broadcast 192.168.58.255
        inet6 fe80::c915:ccd3:2344:e640  prefixlen 64  scopeid 0x20<link>
        ether b8:27:eb:a9:31:f1  txqueuelen 1000  (Ethernet)
        RX packets 86  bytes 9532 (9.3 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 100  bytes 15824 (15.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figura 29 – Raspberry Caso 2
Fuente: Propia

Se arranca el controlador ejecutando la aplicación simple_switch_rest_13.py

```
root@SDN:/usr/local/lib/python3.6/dist-packages/ryu/app# ryu run simple_switch_rest_13.py
loading app simple_switch_rest_13.py
loading app ryu.controller.ofp_handler
creating context wsgi
instantiating app simple_switch_rest_13.py of SimpleSwitchRest13
instantiating app ryu.controller.ofp_handler of OFPHandler
(1055) wsgi starting up on http://0.0.0.0:8080
```

Figura 30 – Controlador SDN Caso 2
Fuente: Propia

Se realizan capturas en los siguientes enlaces: (marcado con una lupa)

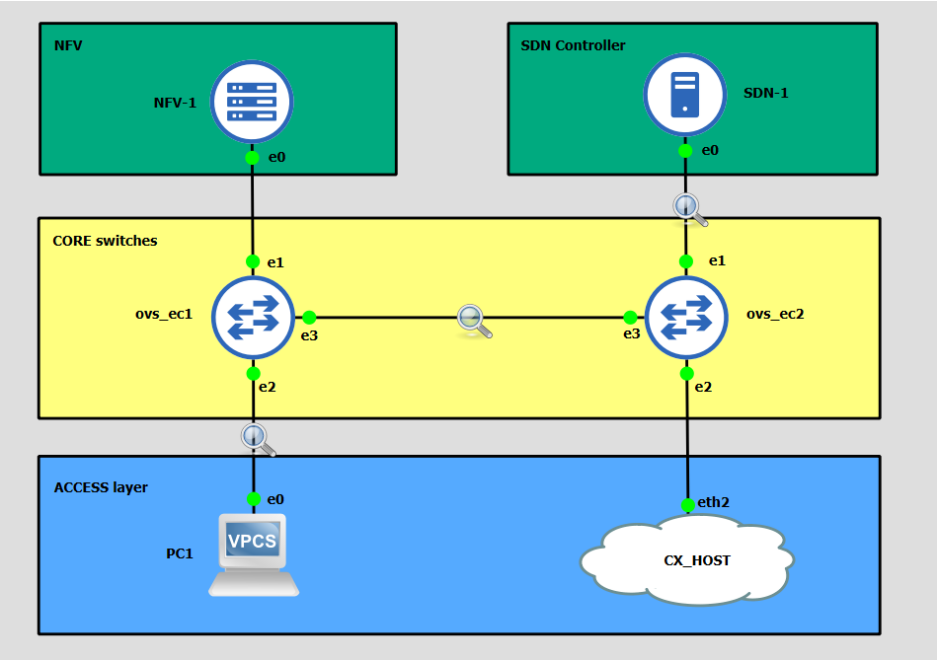


Figura 31 – Capturas Caso 2
Fuente: Propia

Se ejecuta un ping desde el PC hasta la Raspberry:

```
PC1> ping 192.168.58.197
84 bytes from 192.168.58.197 icmp_seq=1 ttl=64 time=86.696 ms
84 bytes from 192.168.58.197 icmp_seq=2 ttl=64 time=6.510 ms
84 bytes from 192.168.58.197 icmp_seq=3 ttl=64 time=6.759 ms
84 bytes from 192.168.58.197 icmp_seq=4 ttl=64 time=7.857 ms
84 bytes from 192.168.58.197 icmp_seq=5 ttl=64 time=6.748 ms
```

Figura 32 – Prueba de conectividad Caso 2
Fuente: Propia

Con esto se comprueba que hay conectividad entre los dispositivos.

Se comprueban los flujos instalados en los switches de manera individualizada. Debido a otros flujos irrelevantes para este caso se filtra por MAC.

```
root@ovs_ec1:/home/sergio# ovs-ofctl dump-flows sw0 | grep 00:50:79:66:68:00
 cookie=0x0, duration=181.881s, table=0, n_packets=0, n_bytes=0, idle_age=181, priority=1,in_port=3,dl_src=00:50:79:66:68:00,dl_dst=52:54:00:50:aa:81 actions=output:2
 cookie=0x0, duration=181.795s, table=0, n_packets=0, n_bytes=0, idle_age=181, priority=1,in_port=2,dl_src=52:54:00:50:aa:81,dl_dst=00:50:79:66:68:00 actions=output:3
 cookie=0x0, duration=154.199s, table=0, n_packets=12, n_bytes=1100, idle_age=142, priority=1,in_port=4,dl_src=b8:27:eb:a9:31:f1,dl_dst=00:50:79:66:68:00 actions=output:3
 cookie=0x0, duration=154.195s, table=0, n_packets=10, n_bytes=942, idle_age=142, priority=1,in_port=3,dl_src=00:50:79:66:68:00,dl_dst=b8:27:eb:a9:31:f1 actions=output:4
```

Figura 33 – Flujos SW 1 Caso 2
Fuente: Propia

```
root@ovs_ec2:/home/sergio# ovs-ofctl dump-flows sw0 | grep 00:50:79:66:68:00
 cookie=0x0, duration=175.205s, table=0, n_packets=12, n_bytes=1100, idle_age=163, priority=1,in_port=3,dl_src=b8:27:eb:a9:31:f1,dl_dst=00:50:79:66:68:00 actions=output:4
 cookie=0x0, duration=175.179s, table=0, n_packets=10, n_bytes=942, idle_age=163, priority=1,in_port=4,dl_src=00:50:79:66:68:00,dl_dst=b8:27:eb:a9:31:f1 actions=output:3
```

Figura 34 – Flujos SW 2 Caso 2
Fuente: Propia

Como se puede ver los flujos son acordes al gráfico:

SWITCH	ORIGEN	DESTINO	ACCIÓN
1	00:50:79:66:68:00	B8:27:EB:A9:31:F1	Re-enviar por el puerto 3
1	B8:27:EB:A9:31:F1	00:50:79:66:68:00	Re-enviar por el puerto 2
2	00:50:79:66:68:00	B8:27:EB:A9:31:F1	Re-enviar por el puerto 2
2	B8:27:EB:A9:31:F1	00:50:79:66:68:00	Re-enviar por el puerto 3

En los flujos aparece el número de puerto +1 porque la cuenta del número de puerto no empieza desde 0, sino desde 1.

Se comprueban las capturas de Wireshark para verificar todo lo anterior:

Ping exitoso entre el equipo y la raspberry

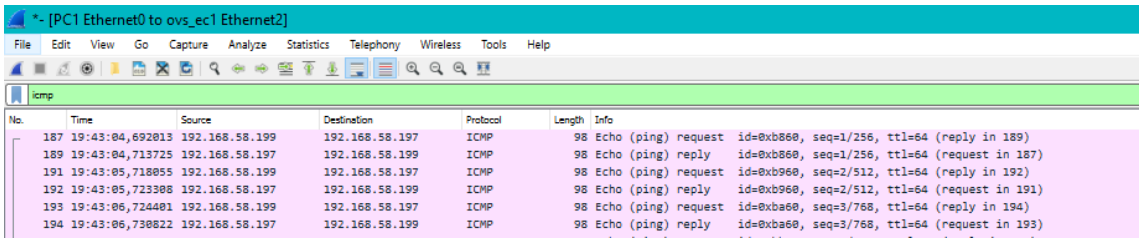


Figura 35 – Captura enlace PC al switch 1
Fuente: Propia

El switch 1 indica al controlador mediante mensaje OpenFlow (OFPT_PACKET_IN) que ha recibido un paquete ICMP con origen 192.168.58.199 y destino 192.168.58.197

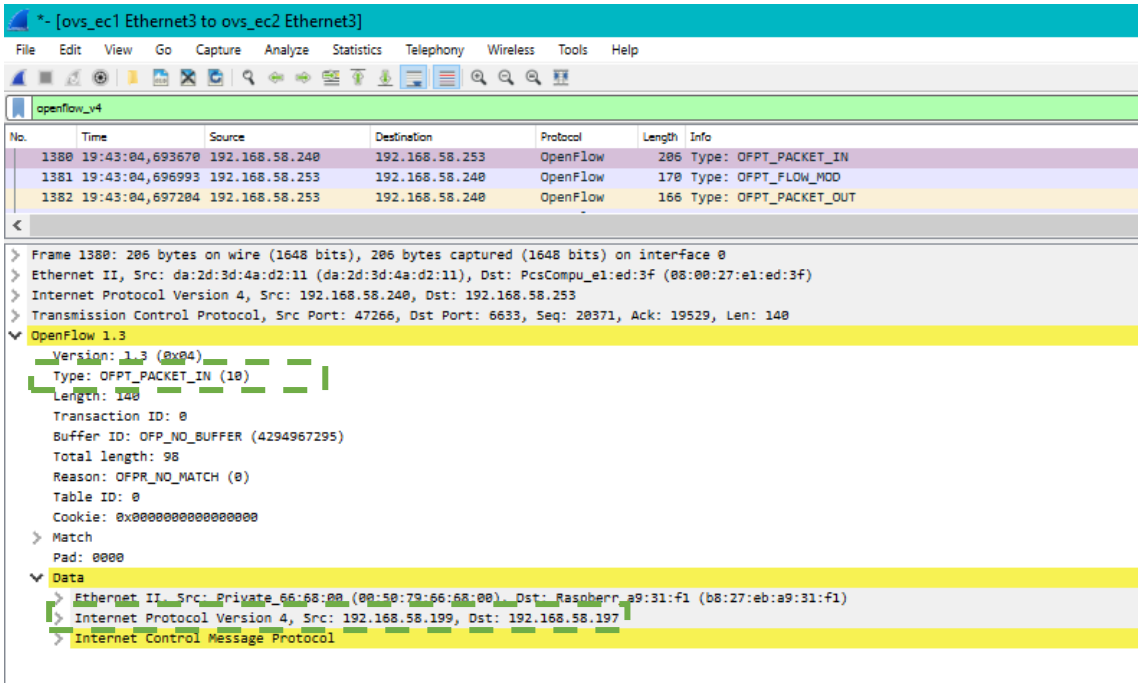


Figura 36 – Captura enlace SW 1 y SW 2 OFPT IN
Fuente: Propia

El controlador le indica al switch que debe re-enviar el paquete por el puerto 4 (OFPT_PACKET_OUT)

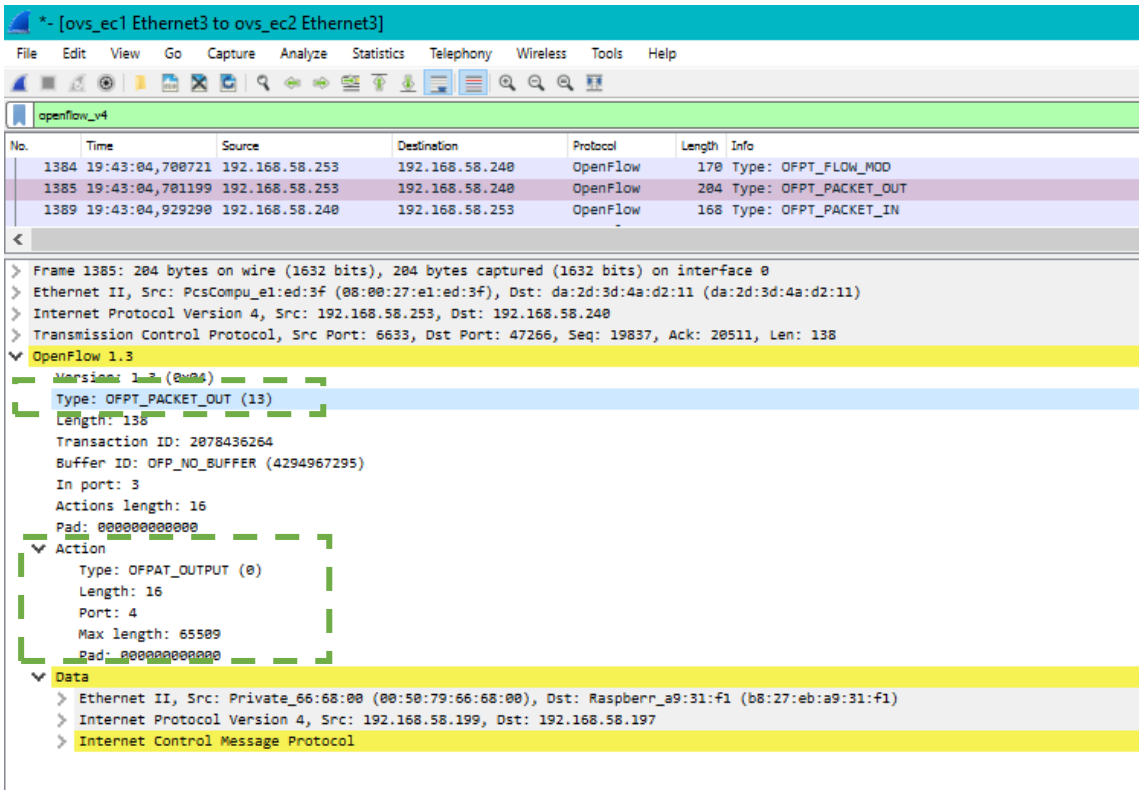


Figura 37 - Captura enlace SW 1 y SW 2 OFPT OUT
Fuente: Propia

Mediante la aplicación switch_management.py se obtiene la tabla de MACs de los switches.

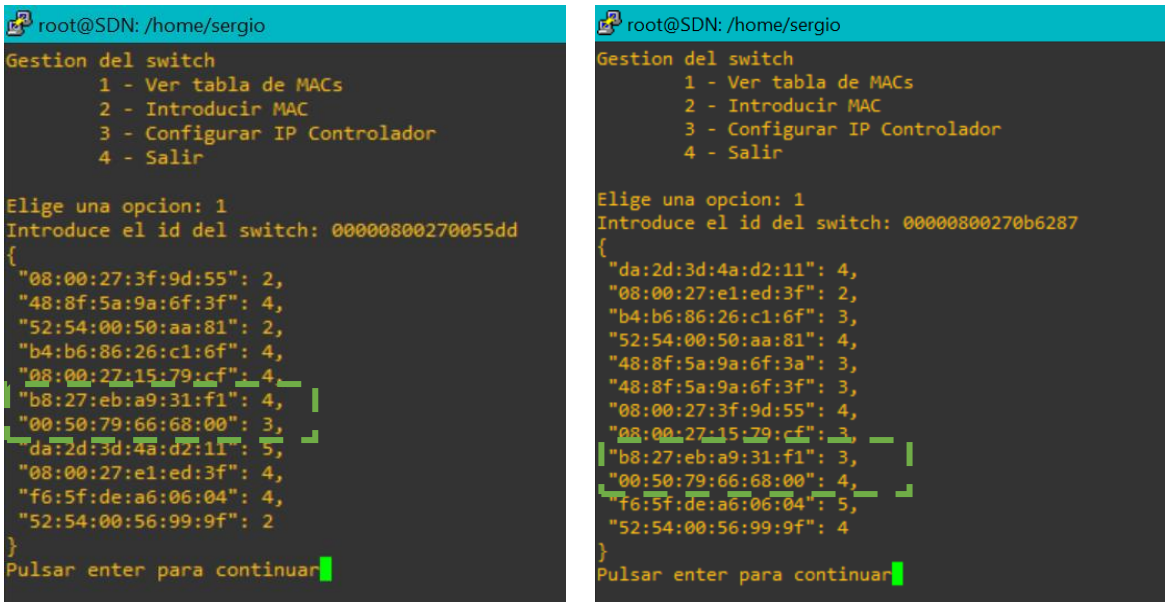


Figura 38 – Tablas de MACs de los switches
Fuente: Propia

Como se puede ver la tabla de MACs concuerda con los puertos a los que están conectados los equipos.
Con esto se comprueban todos los puntos marcados como objetivos para el caso 2.

4.2.3 Caso de uso 3

Para este caso de uso se conecta un equipo por WiFi y otro equipo por cable. El equema a seguir en este caso es:

- Comprobar que hay comunicación entre los dos dispositivos conectados.
- Aplicar el Firewall en los switches mediante una aplicación.
- Incluir reglas en el Firewall mediante una aplicación para impedir el tráfico entre los dos dispositivos.
- Comprobar que deja de haber comunicación entre los dos dispositivos conectados.
- Comprobar los mensajes OpenFlow que se envían entre el controlador y los switches.

Para comprobar los objetivos definidos en el caso de uso 3 es necesario:

- Verificar mediante protocolo ICMP que hay conectividad de red entre los dos dispositivos.
- Verificar que tras aplicar el Firewall en los switches ya no hay conectividad mediante ICMP.
- Verificar mediante línea de comandos los flujos instalados en cada switch relativos a la comunicación entre dispositivos.
- Realizar y estudiar capturas de Wireshark en los enlaces relevantes de la maqueta para ver los paquetes OpenFlow desde y hacia el controlador.

La arquitectura de red para este caso es la siguiente:

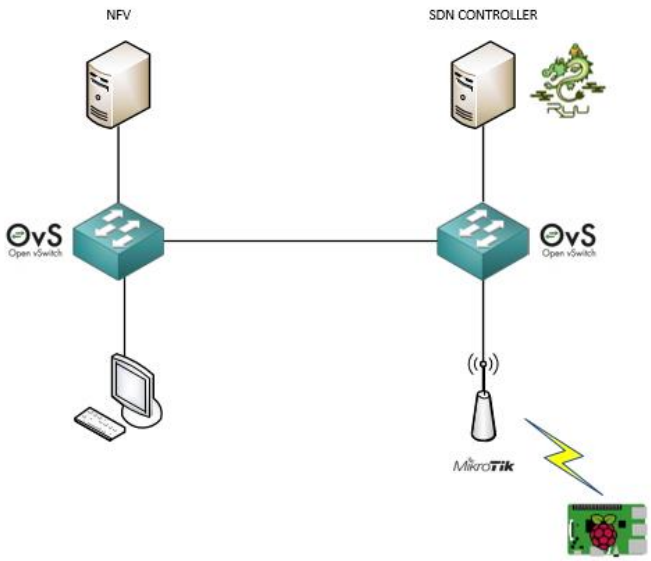


Figura 39 – Arquitectura caso 3
Fuente: Propia

Para este caso la arquitectura de red propuesta es la misma que la del caso 2.

Se configura la red para ambos dispositivos involucrados.

Red para el PC:

```
PC1> show ip IP 192.168.58.196/24

NAME        : PC1[1]
IP/MASK     : 192.168.58.196/24
GATEWAY     : 0.0.0.0
DNS         :
DHCP SERVER : 192.168.58.250
DHCP LEASE  : 595, 600/300/525
MAC         : 00:50:79:66:68:00
LPORT       : 10027
RHOST:PORT  : 127.0.0.1:10028
MTU:        : 1500
```

Figura 40 – PC1 caso 3
Fuente: Propia

Red para la Raspberry Pi:

```
pi@raspberrypi: ~
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.58.197  netmask 255.255.255.0  broadcast 192.168.58.255
        inet6 fe80::c915:ccd3:2344:e640  prefixlen 64  scopeid 0x20<link>
        ether b8:27:eb:a9:31:f1  txqueuelen 1000  (Ethernet)
        RX packets 383  bytes 74489 (72.7 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 137  bytes 21309 (20.8 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figura 41 – Raspberry caso 3
Fuente: Propia

Para este caso las capturas se hacen en los siguientes enlaces (marcado con la lupa)

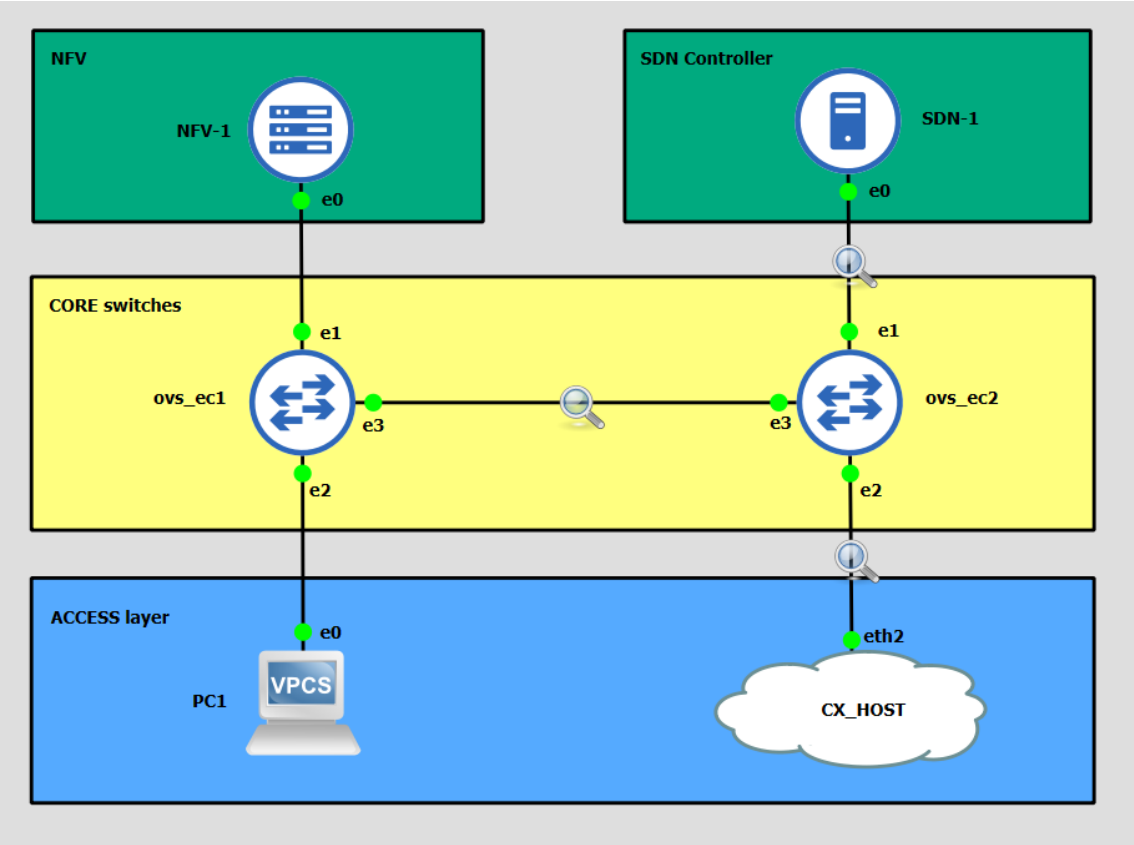


Figura 42 – Capturas caso 3
Fuente: Propia

Se arranca el controlador ejecutando la aplicación rest_firewall.py

```
root@SDN:/usr/local/lib/python3.6/dist-packages/ryu/app# ryu run rest_firewall.py
loading app rest_firewall.py
loading app ryu.controller.ofp_handler
instantiating app None of DPSet
creating context dpset
creating context wsgi
instantiating app rest_firewall.py of RestFirewallAPI
instantiating app ryu.controller.ofp_handler of OFPHandler
(1497) wsgi starting up on http://0.0.0.0:8080
[FW][INFO] dpid=00000800270055dd: Join as firewall.
[FW][INFO] dpid=00000800270b6287: Join as firewall.
```

Figura 43 – Controlador caso 3
Fuente: Propia

Como se puede ver los dos switches se unen como firewall, pero se va a actuar sobre el switch 2 porque es suficiente.

Se ejecuta la aplicación de gestión de firewall y se consultan las reglas aplicadas en el switch 2

Figura 44 – Reglas del firewall switch 2
Fuente: Propia

La única regla que hay es para permitir todo el tráfico, por lo que la conectividad funciona sin problema.

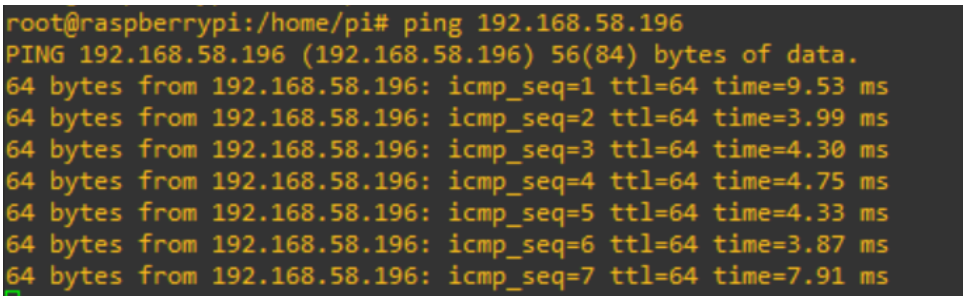


Figura 45 – Comprobación de conectividad Caso 3
Fuente: Propia

Mediante la aplicación se introduce una regla para denegar el tráfico. La regla se introduce en ambos sentidos para que se deniegue tanto la ida como la vuelta del tráfico.

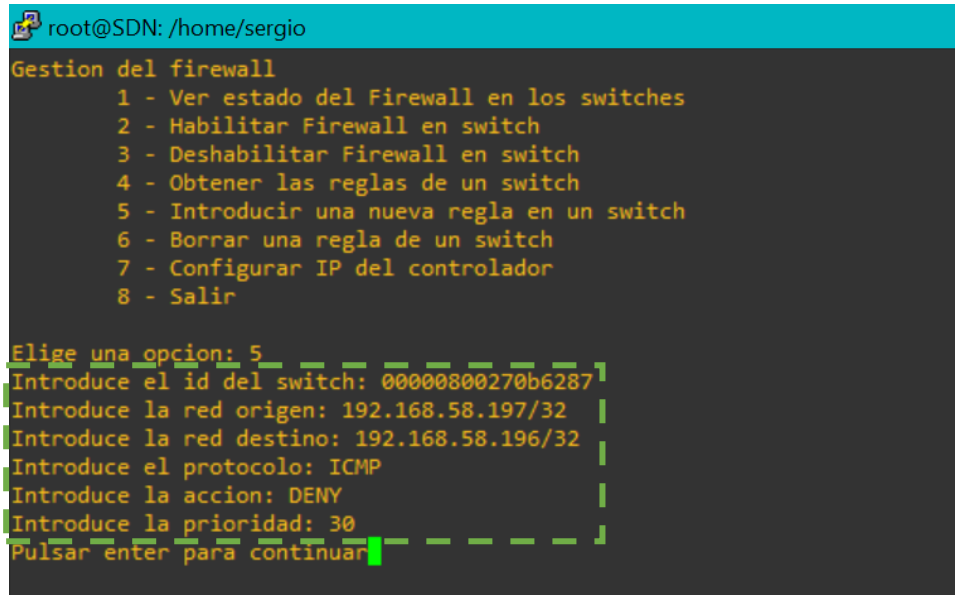


Figura 46 – Regla de ida caso 3
Fuente: Propia

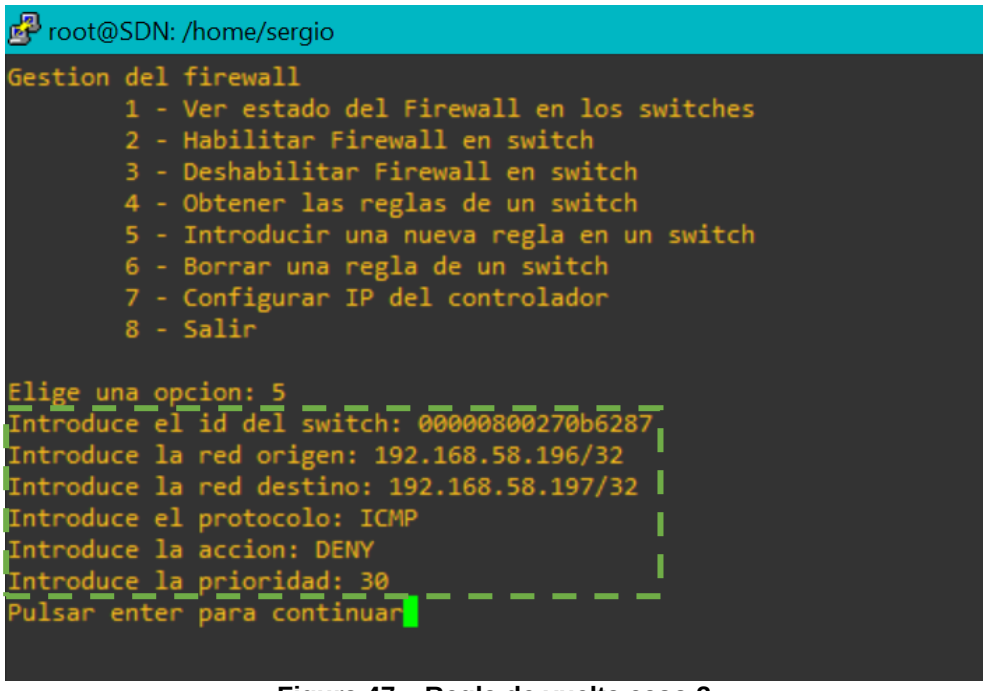


Figura 47 – Regla de vuelta caso 3
Fuente: Propia

De nuevo, a través de la aplicación se comprueba que se han introducido las reglas en los equipos:

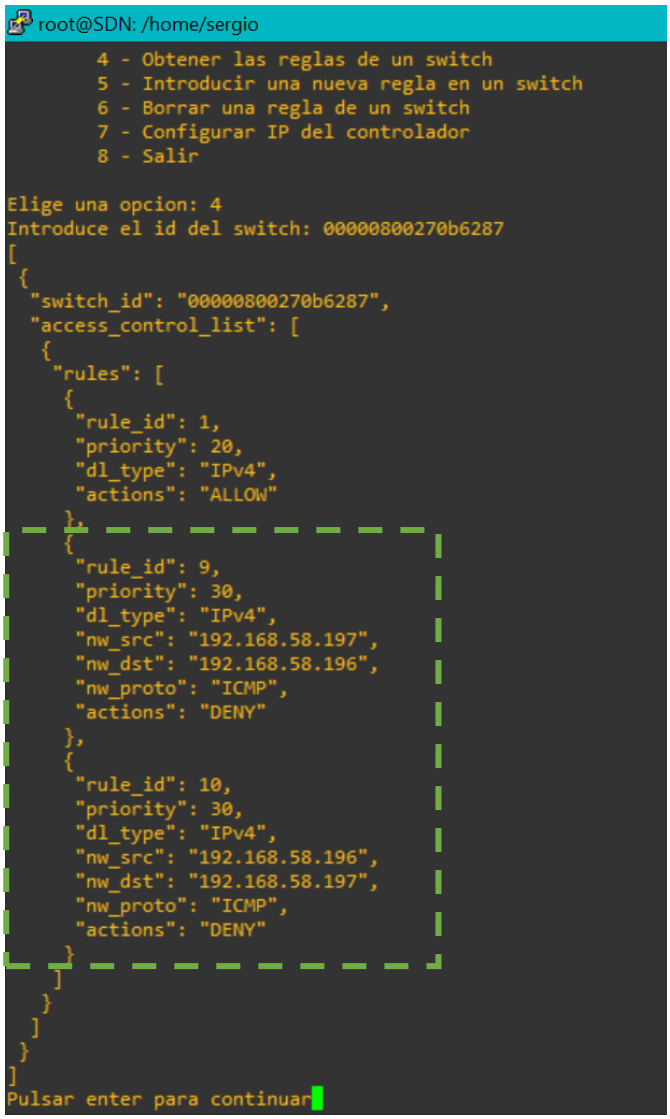


Figura 48 – Reglas switch 2 caso 3
Fuente: Propia

Una vez que se ha comprobado que las reglas están en el firewall del equipo se comprueban los flujos instalados en el switch.

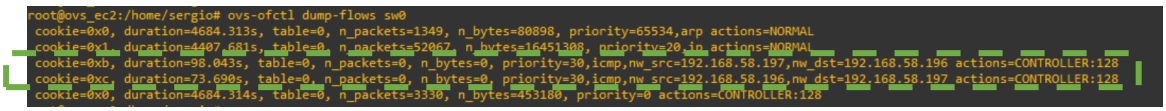


Figura 49 – Flujos switch 2 caso 3
Fuente: Propia

Hay dos flujos que indican que cuando llegue el tráfico correspondiente a las reglas se mande al controlador para decidir qué hacer con él.

Se comprueba de nuevo la conectividad:

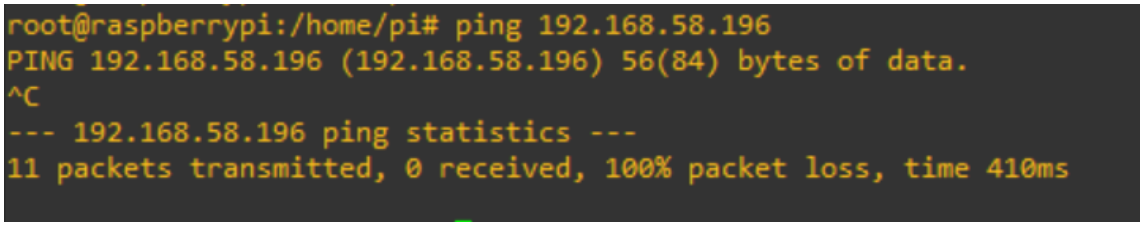


Figura 50 – Comprobación conectividad tras aplicar reglas caso 3
Fuente: Propia

Ahora no hay conectividad entre los equipos y se ve que el controlador lo está bloqueando:

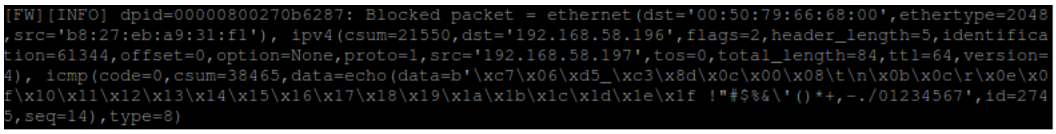


Figura 51 – Bloqueo tráfico controlador
Fuente: Propia

Una vez visto que el tráfico se bloquea se eliminan las dos reglas introducidas a través de la aplicación para que vuelva a haber conectividad.

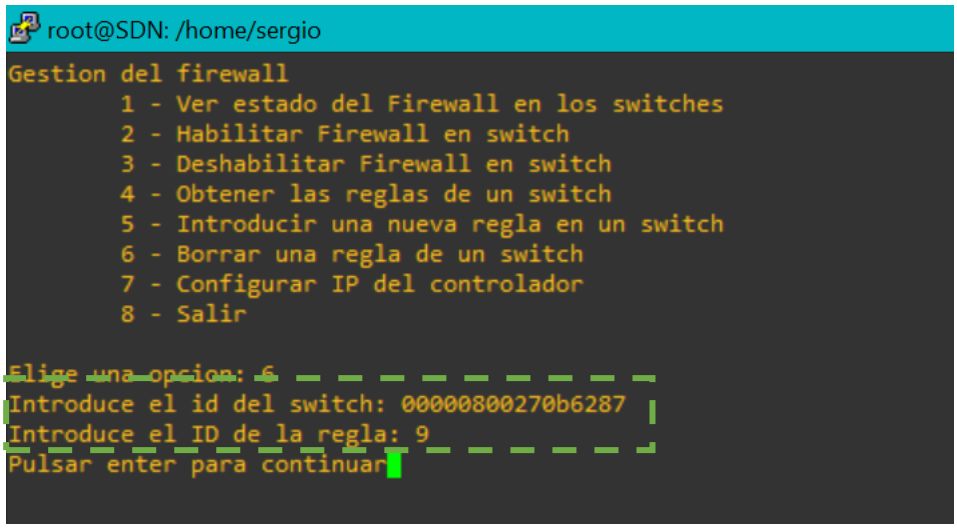


Figura 52 – Eliminación regla 1 caso 3
Fuente: Propia

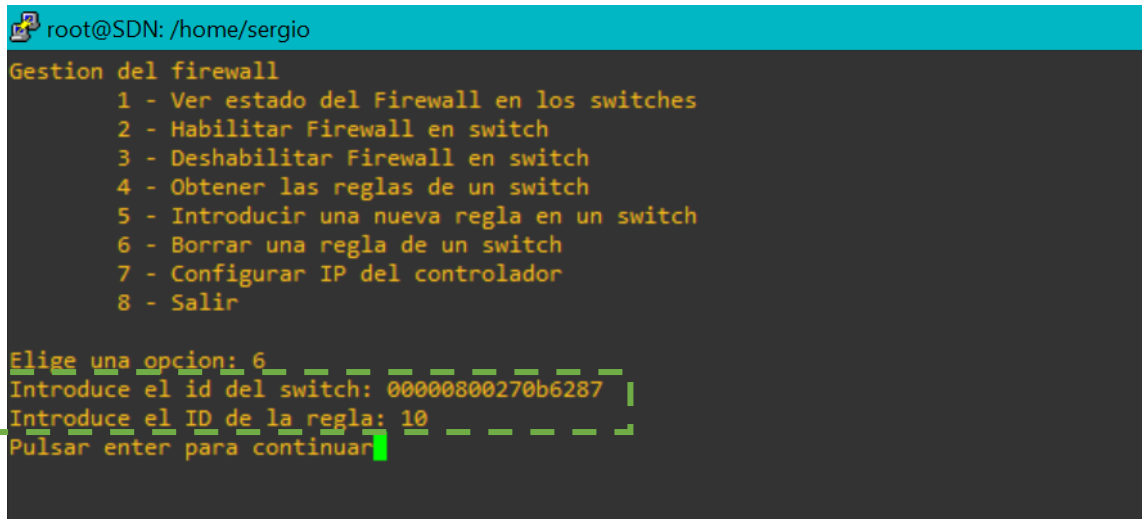


Figura 53 – Eliminación regla 2 caso 3
Fuente: Propia

Se comprueba que las reglas ya no están en el switch:

Figura 54 – Comprobación eliminación de reglas caso 3
Fuente: Propia

También se comprueba que los flujos instalados en los switches ya no están:

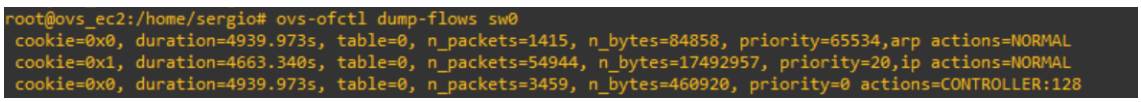


Figura 55 – Comprobación eliminación de flujos caso 3
Fuente: Propia

De nuevo se prueba la conectividad entre los equipos para verificar que ya funciona de nuevo:

```
root@raspberrypi:/home/pi# ping 192.168.58.196
PING 192.168.58.196 (192.168.58.196) 56(84) bytes of data.
64 bytes from 192.168.58.196: icmp_seq=176 ttl=64 time=4.14 ms
64 bytes from 192.168.58.196: icmp_seq=177 ttl=64 time=4.63 ms
64 bytes from 192.168.58.196: icmp_seq=178 ttl=64 time=4.08 ms
64 bytes from 192.168.58.196: icmp_seq=179 ttl=64 time=4.29 ms
64 bytes from 192.168.58.196: icmp_seq=180 ttl=64 time=7.88 ms
64 bytes from 192.168.58.196: icmp_seq=181 ttl=64 time=6.07 ms
```

Figura 56 – Comprobación conectividad caso 3
Fuente: Propia

Una vez visto el funcionamiento de los equipos se comprueban las capturas de Wireshark.

Inicialmente hay conectividad por lo que el ping funciona sin problema, hay tanto paquetes icmp request como icmp reply.

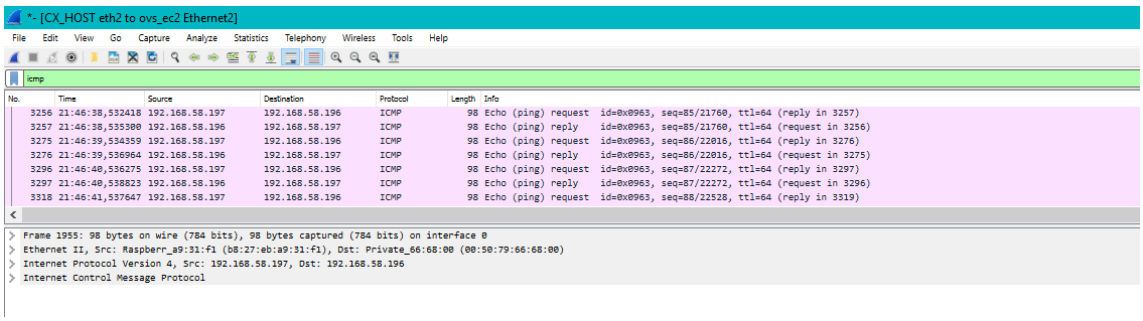


Figura 57 – Captura 1 caso 3
Fuente: Propia

En el momento que se aplican las reglas deja de haber conectividad, solo hay paquetes icmp request, pero ninguno de reply.

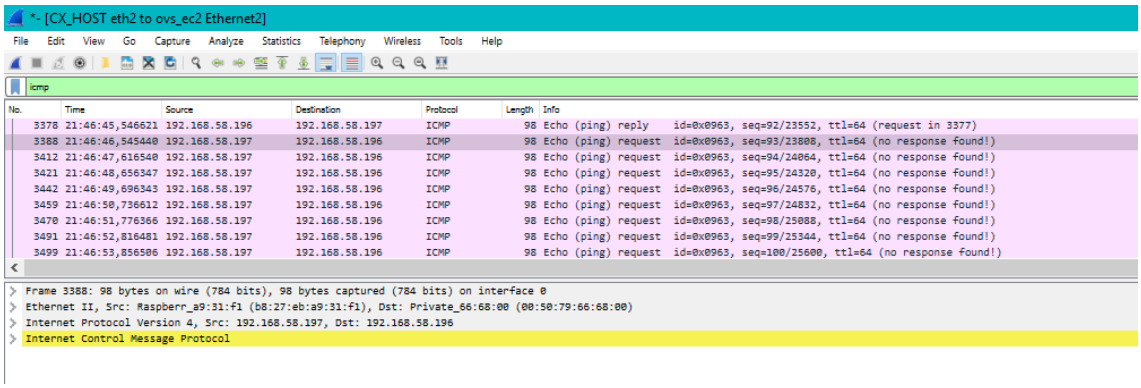


Figura 58 – Captura 2 caso 3
Fuente: Propia

El controlador inserta el flujo relativo a las reglas en el switch 2 indicando que los paquetes han de ser re-enviados al mismo (OFPT_FLOW_MOD)

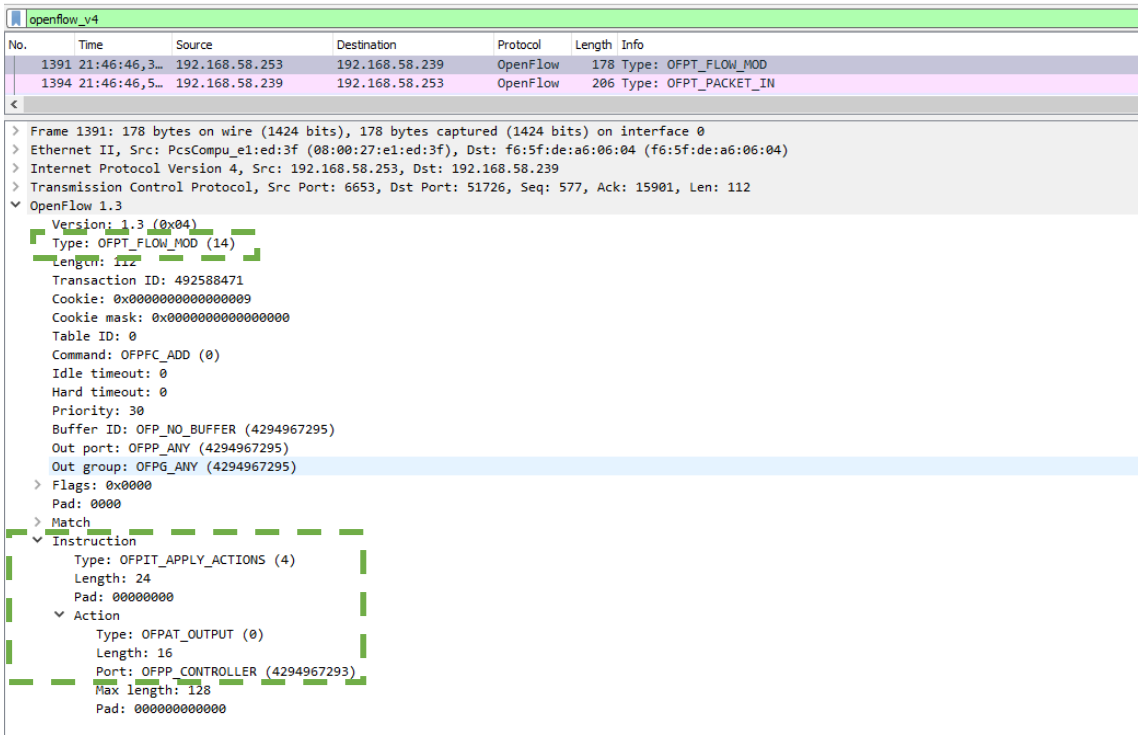


Figura 59 – Captura 3 caso 3
Fuente: Propia

El switch manda el tráfico entrante al controlador para que lo bloquee (OFPT_PACKET_IN)

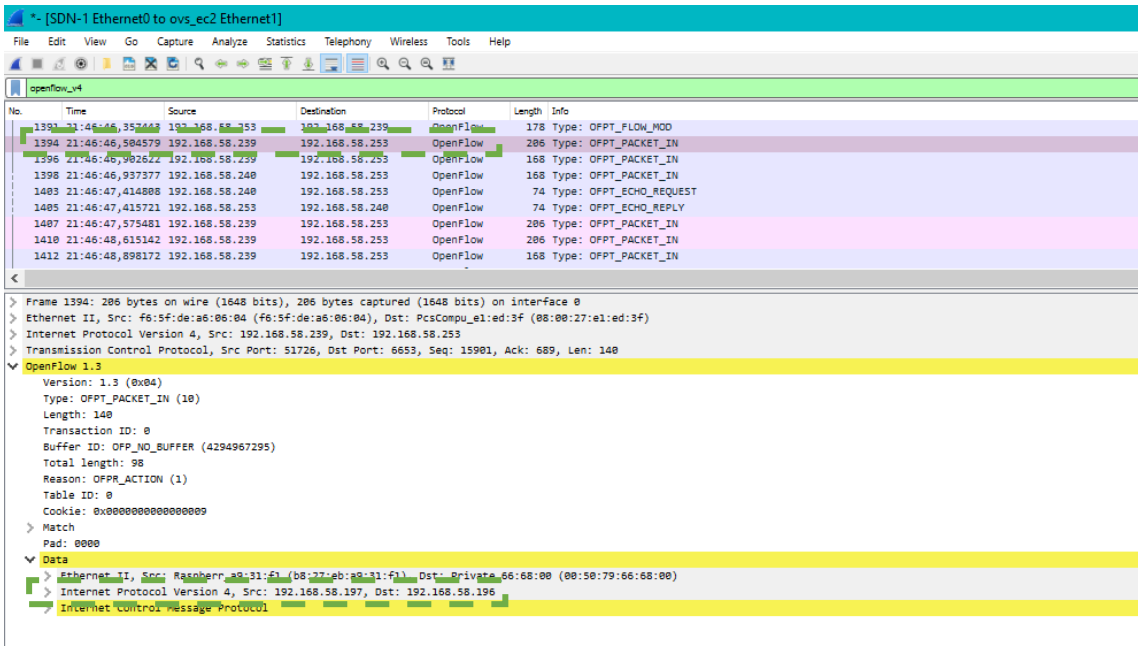


Figura 60 – Captura 4 caso 3
Fuente: Propia

Se eliminan las reglas, por lo que vuelve a haber conectividad, hay tanto paquetes icmp request como icmp reply de nuevo.

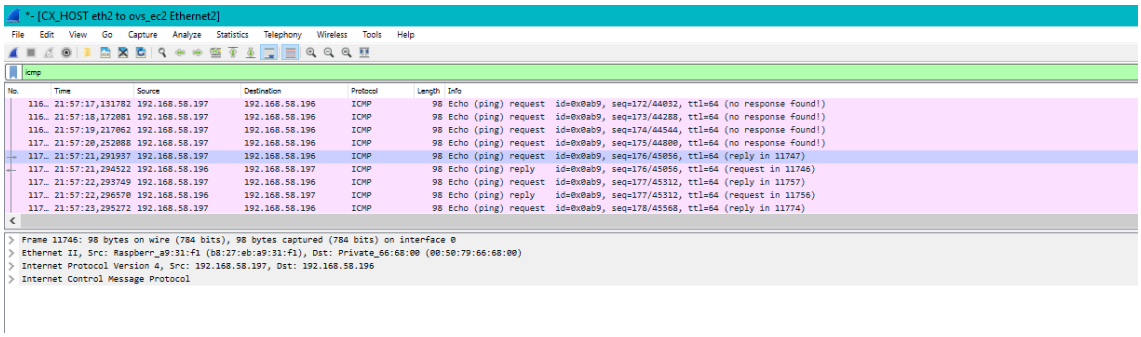


Figura 61 – Captura 5 caso 3
Fuente: Propia

El switch manda de nuevo los paquetes al controlador para ver que hacer con ellos.

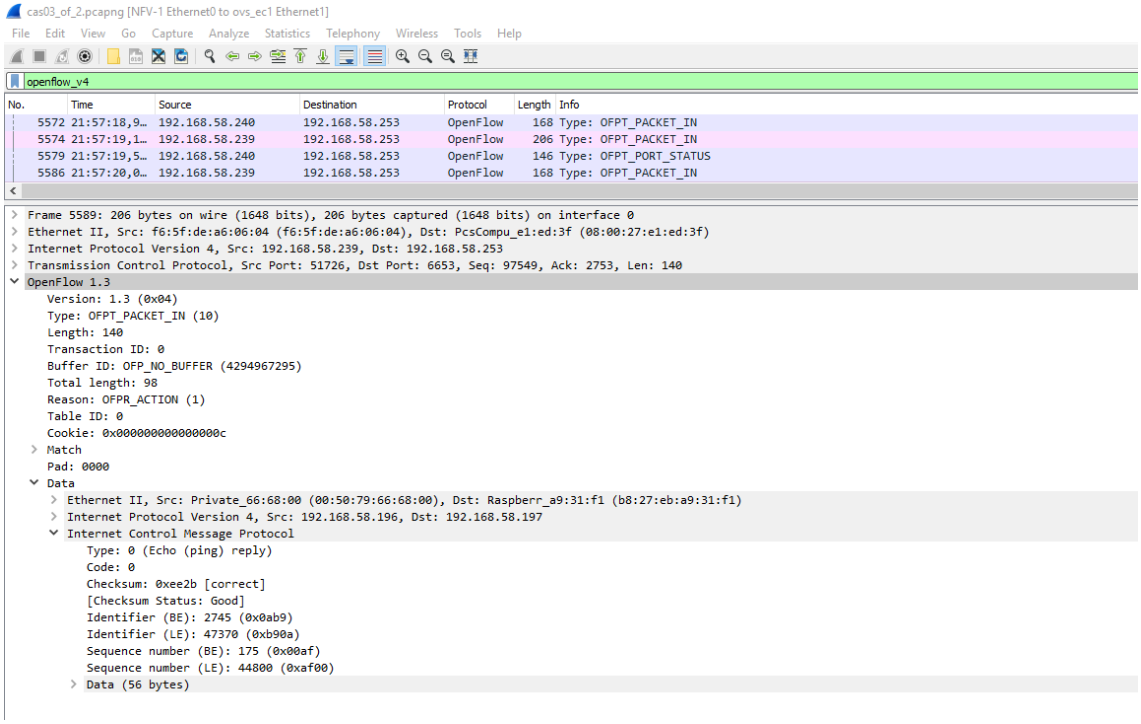


Figura 62 – Captura 6 caso 3
Fuente: Propia

El controlador elimina el flujo instalado anteriormente en el switch, por lo que ya no hay problemas en la conectividad

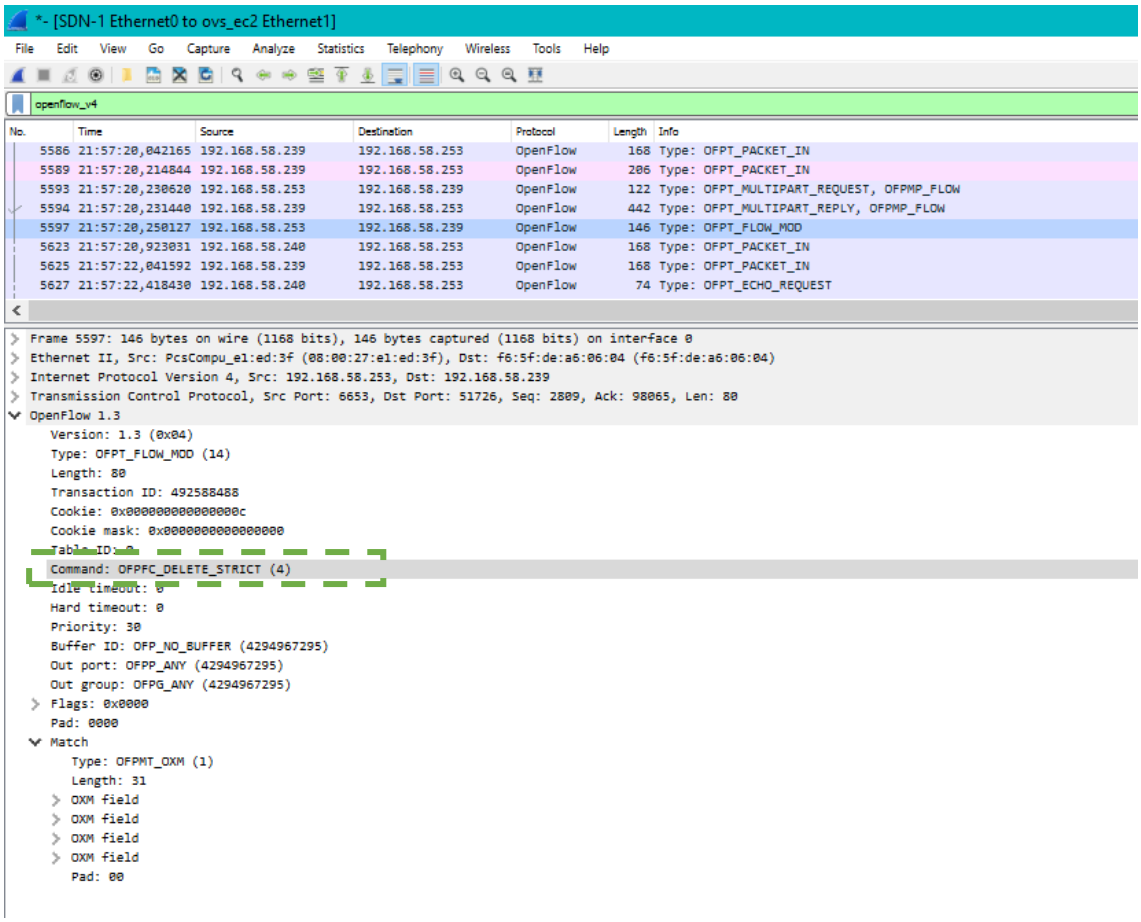


Figura 63 – Captura 7 caso 3
Fuente: Propia

Con esto se comprueban todos los puntos marcados como objetivos para el caso 3.

5. Conclusiones

Este trabajo ha permitido integrar SDN dentro de una red híbrida (cable + WiFi). Adicionalmente, el trabajo ha permitido estudiar y elegir la mejor manera de gestionar el tráfico inalámbrico dentro de la red. El uso de SDN junto con una red WiFi no está tan extendido en comparación con el uso de SDN en una red cableada tradicional, por lo que ha sido de gran interés realizar esta integración. A parte, dentro del ámbito SDN el trabajo también ha permitido explorar la manera de integrar en la red NFVs, lo cual tiene una gran relevancia, puesto que la tendencia actual es la de intentar virtualizar equipos como funciones de red para reducir los costes asociados a los dispositivos físicos.

Por otro lado el trabajo ha permitido profundizar en el nuevo estándar inalámbrico y comprobar las innumerables ventajas que tiene para dispositivos IoT que necesitan conectividad. El estándar es relativamente nuevo y el número de dispositivos IoT no deja de crecer, por lo que es de gran interés conocer las funcionalidades y ventajas que aporta el estándar a este tipo de dispositivos.

Por último, durante el trabajo se han realizado varias aplicaciones para simplificar la gestión de la red demostrando que es más sencillo gestionar una red SDN frente a una red tradicional.

Se considera que se han cumplido los objetivos marcados en el primer capítulo puesto que se ha conseguido aplicar con éxito SDN a una red inalámbrica. En este caso concreto se ha simulado la red de un entorno industrial contando con dispositivos conectados por WiFi y por cable. Se ha conseguido virtualizar dos funciones de red necesarias para cualquier red WiFi dentro de un entorno empresarial, la controladora WiFi y el servidor Radius, posibilitando el poder hacer despliegues de manera sencilla donde sea necesario. Por último se han comprobado cuales son las ventajas que proporciona el nuevo estándar WiFi a los dispositivos IoT.
Una vez realizados todos los estudios se ha diseñado una arquitectura de red integrando todos los elementos que podría extenderse y ampliarse en cualquier entorno ya sea industrial o no.
Se han propuesto tres casos de uso que son comunes dentro de cualquier entorno y han servido para comprobar la validez de la arquitectura de red propuesta.

En cuanto a la planificación a nivel de tiempos no ha habido desviaciones reseñables, si bien es cierto que ha sido necesario cambiar ciertas herramientas propuestas por otras por problema de la capacidad del equipo que aloja y simula la maqueta de red. La metodología propuesta se ha mantenido durante el trabajo. El trabajo se ha estructurado en cinco fases, las tres primeras fases son independientes entre sí, pero las dos últimas si tienen dependencia. A pesar de ello, las fases del trabajo se han abordado secuencialmente y sin mayor problema.

Por el lado de las líneas futuras que no se exploran en el presente trabajo queda pendiente el aspecto de la movilidad de clientes. Este trabajo se ha centrado en la asociación de clientes a la red WiFi y a la gestión de los flujos. Puede ser necesario el uso de aplicaciones adicionales y de comunicación entre el controlador SDN y la controladora WiFi para garantizar la movilidad de clientes entre APs.

6. Glosario

5G – Quinta generación de redes móviles
AAA – Autenticación, autorización y auditoría, protocolo de seguridad en redes IP
AP – Punto de acceso a la red
API – Interfaz de programación avanzada
ASAv – Apliance de seguridad avanzada virtual de Cisco
BSS – Sistema de soporte al negocio
CA – Protocolo para evitar colisiones en redes inalámbricas
CAPWAP – Protocolo de control y aprovisionamiento de APs
CNC – Control numérico
CPD – Centro de procesamiento de datos
CSMA – Protocolo de control de acceso múltiple con escucha de portadora
DHCP – Protocolo de configuración dinámica de equipos
EAP – Protocolo de autenticación extensible para redes inalámbricas
ETSI – Instituto europeo de normas de telecomunicaciones
gRPC – Llamada de procedimiento remoto
HP – Hewlet Packard, fabricante de equipos
ICMP – Protocolo de mensajes de control en internet
IEEE – Instituto de ingenieros eléctricos y electrónicos
IoT – Internet de las cosas
ISRv – Router virtual de servicios integrados de Cisco
IP – Protocolo de internet
ISP – Proveedor de servicios de internet
KVM – Máquina virtual de Linux
LTE – Evolución a largo plazo, cuarta generación de redes inalámbricas
LVAP – Punto de acceso virtual ligero
MAC – Control de acceso al medio
MANO – Gestión y orquestación de NFV de código abierto
MIMO – Entradas múltiples y salidas múltiples
MPLS – Conmutación de etiquetas multiprotocolo
MU-MIMO - Entradas múltiples y salidas múltiples multi usuario
NETCONF – Protocolo de configuración de red
NFV – Virtualización de funciones de red
NFVI – Virtualización de infraestructura de funciones de red
NFVO – Orquestador de funciones virtuales de red
NGFW – Firewall de nueva generación
OFDM – Multiplexación por división en frecuencia ortogonal
OFDMA – Acceso múltiple por división en frecuencia ortogonal
ODL – OpenDayLight, controlador SDN
ONAP – Plataforma de automatización de la red abierta
ONOS – Sistema operativo de red abierto, controlador SDN
Open-O – Orquestador abierto
OPNFV – Plataforma de virtualización de funciones de red abierta.
OSS – Sistema de soporte a las operaciones
OvS – Open vSwitch
PC – Ordenador personal
PLC – Controlador lógico programable

QAM – Modulación de amplitud en cuadratura
QCOW2 – Copia en escritura de QUEMU, extensión de discos virtuales en KVM
QoS – Calidad de servicio
REST – Transferencia de estado representacional, arquitectura de software
SDN – Redes definidas por software
SD-WAN – Red de área extensa definida por software
SD-WLAN – Red local inalámbrica definida por software
SISO – Entrada individual y salida individual
SNMP – Protocolo de gestión simple de red
SSID – Identificador de set de servicios
UTP – Par trenzado no blindado
VIM – Gestor de infraestructura virtual
VLAN – Red de área local virtual
VNF – Función de red virtual
VNFM – Gestor de funciones de red virtuales
vWAAS – Servicios de aplicación de área amplia de Cisco
WAN – Red de área amplia
WiFi – Wireless Fidelity, protocolo de acceso inalámbrico a la red
WLAN – Red de área local inalámbrica
WLC – Controladora de redes locales

7. Bibliografía

[1] Historia de SDN.
Ref: https://openzen.wordpress.com/2015/02/12/historia-del-sdn/

[2] Redes de nueva generación, Protocolo OpenFlow. Autor: Daniel Morató Osés, Uniersidad de Navarra.

[3] Comparison of software defined networking controllers. Autor: Farzaneh Pakzad.

[4] OPNFV.
Ref: https://wiki.opnfv.org/

[5] KVM.
Ref: https://www.redhat.com/es/topics/virtualization/what-is-KVM

[6] Docker.
Ref:https://docs.microsoft.com/es-es/dotnet/architecture/containerized-lifecycle/what-is-docker

[7] El concepto de OpenStack.
Ref: https://www.redhat.com/es/topics/openstack

[8] VMware Joins OPEN-O Project for NFV Orchestration. Autor: David Ramel

[9] El laboratorio de referencia de NFV de Telefónica libera el stack de orquestación NFV OpenMANO.
Ref: https://www.telefonica.com/es/web/sala-de-prensa/-/el-laboratorio-de-referencia-nfv-de-telefonica-libera-el-stack-de-orquestacion-nfv-openmano

[10] What is JuJu.
Ref: https://juju.is/docs/what-is-juju

[11] RIFT.io.
Ref: https://riftio.com/about-us/

[12] IEEE 802.11 Standard.

[13] 6 Ways 802.11ax Gives You a Better Wi-Fi Experience.
Ref: https://blogs.arubanetworks.com/solutions/6-ways-802-11ax-gives-you-better-wi-fi-experience/

[14] OFDMA. Autor 3Cu Electrónica.
Ref: https://sites.google.com/site/3cuelectronica/home/multiplexacion/ofdma

[15] How does BSS Coloring Work In 802.11ax?. Autor: David Coleman
Ref: https://www.extremenetworks.com/extreme-networks-blog/how-does-bss-coloring-work-in-802-11ax/

[16] El estándar IEEE 802.11 Wireless LAN. Autor: Francisco López Ortiz, Universidad de Navarra.

[17] Cisco – Wireless LAN Controllers
Ref: https://www.cisco.com/c/en/us/products/wireless/wireless-lan-controller/index.html

[18] Aruba Gateways and Controllers
Ref: https://www.arubanetworks.com/products/wireless/gateways-and-controllers/

[19] Control And Provisioning of Wireless Access Points (CAPWAP) Protocol Specification. Autores: P. Calhoun, Ed., Cisco Systems, Inc., M. Montemurro, Ed., Research In Motion, D. Stanley, Ed., Aruba Networks

[20] Chandelle: Smooth and Fast WiFi Roaming with SDN/OpenFlow. Autores: Sergey Monin, Alexander Shalimov, Rusian. L. Smeliansky

[21] Overview of Different Approaches for Leveraging SDN in Mobile Networks. Autor: Raghunath Deshpande

[22] Cisco Enterprise Network Functions Virtualization.
Ref: https://www.cisco.com/c/en/us/solutions/enterprise-networks/enterprise-network-functions-virtualization-nfv/index.html#~what's-inside

[23] SDN, OpenFlow for WLAN
Ref:https://www.arubanetworks.com/techdocs/Instant_83_WebHelp/Content/Instant_UG/Services/SDN.htm

[24] Mikrotik.
Ref: https://mikrotik.com/aboutus

[25] Open vSwitch.
Ref: https://www.openvswitch.org/

8. Anexos

8.1 Anexo 1: Configuración de red de las máquinas virtuales

Controlador SDN:

```
# ufw disable
# cat /etc/netplan/01-netcfg.yaml
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s3:
      dhcp4: no
      addresses:
        - 192.168.58.253/24
# netplan apply
```

NFV:

```
# virsh net-destroy default
# virsh net-undefine default
# ufw disable
# cat /etc/netplan/01-netcfg.yaml
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s3:
      dhcp4: no
  bridges:
    br0:
      interfaces: [enp0s3]
      addresses: [192.168.58.252/24]
      dhcp4: no
# netplan apply
```

Freeradius:

```
# ufw disable
# vi /etc/netplan/01-netcfg.yaml
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s3:
      dhcp4: no
      addresses:
        - 192.168.58.251/24
# netplan apply
```

Controladora WiFi:

```
/interface bridge  add name=lan
/interface bridge port add bridge=lan interface=ether1
```

```
/ip address add address=192.168.58.250/24 interface=lan network=192.168.58.0
/ip firewall filter add action=accept chain=forward dst-address=0.0.0.0 src-address=0.0.0.0
/openflow add controllers=192.168.50.253 datapath-id=1/52:54:00:50:AA:81 disabled=no
name=oflow1
```

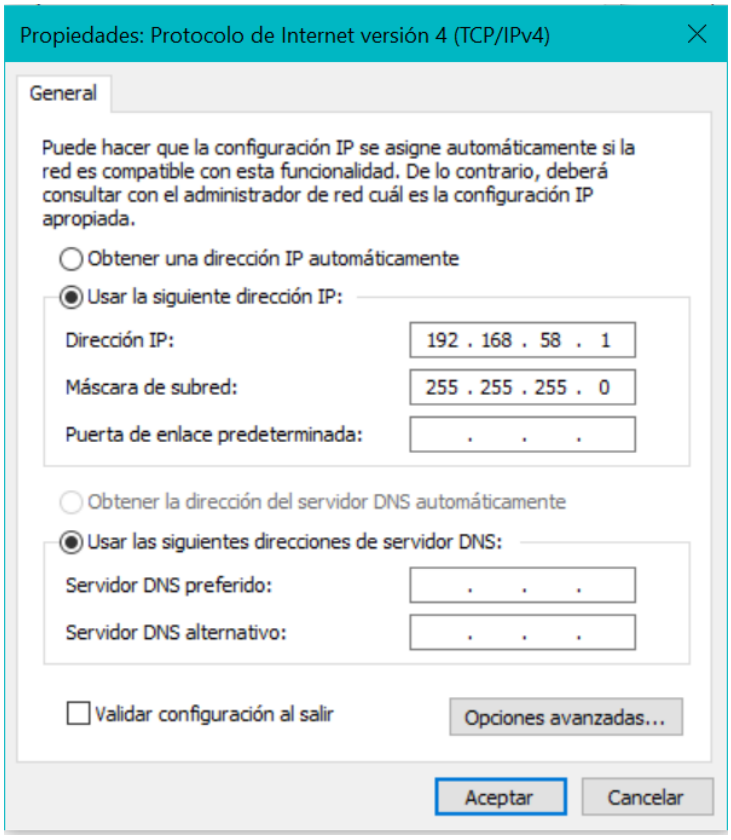
AP WiFi:

```
/interface bridge add name=bridgeTFM
/interface bridge port add bridge=bridgeTFM interface=wlan1
/interface bridge port add bridge=bridgeTFM interface=ether2
/interface bridge port add bridge=bridgeTFM interface=ether3
/ip address add address=192.168.58.229/24 interface=bridgeTFM network=192.168.58.0
/openflow add controllers=192.168.50.253 datapath-id=1/48:8F:5A:9A:6F:39 disabled=no
name=oflow1
```

HOST:

Se configura una IP estática en el equipo:

Panel de control\Redes e Internet\Conexiones de red



8.2 Anexo 2: Aplicaciones

Desplegar NFV

```
#!/bin/bash
```

```
echo "#########################################################"
echo "# DESPLIEGUE DE MAQUINAS VIRTUALES EN KVM               #"
echo "#########################################################"

read -p "Directorio del fichero qcow2: " qcow2_dir
read -p "Nombre de la maquina virtual: " mv_name
read -p "Memoria de la maquina virtual: " mv_ram
reap -p "Numero de CPUs de la maquina virtual: " mv_cpu
read -p "Bridge de red al que conectar la maquina: " mv_net

virt-install --name $mv_name --memory $mv_ram --vcpus=$mv_cpu --disk $qcow2_dir,bus=ide
--import --os-type=generic --os-variant=generic --network bridge=$mv_net
virsh autostart $mv_name
virsh list –all
```

Configurar WLC

```
#!/usr/bin/python3.6
import paramiko
from getpass import getpass

print("#################################################################")
print("# CONFIGURACION DE WLC MIKROTIK                                 #")
print("#                                                               #")
print("# Configuracion por defecto:                                    #")
print("# Tipo de seguridad: WPA2-EAP                                   #")
print("# Encriptacion: AES-CCM                                         #")
print("# Certificados: No                                              #")
print("# Regulacion WiFi: Spain                                        #")
print("#################################################################")

ip_address = input("IP WLC: ")
usuario = input("Usuario: ")
passw = getpass()
pool_dhcp = input("Pool DHCP a configurar: ")
networ_dhcp = input("Red DHCP a configurar: ")
bridge_conf = input("Bridge del equipo: ")
radius_ip = input("IP del servidor Radius: ")
radius_secret = input("Secreto compartido Radius: ")
wifi_ssid = input("SSID WiFi: ")

client = paramiko.SSHClient()
client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
client.connect(hostname=address, username=user, password=passw)
stdin,stdout,stderr = client.exec_command("ip pool add name=pool1 ranges=%s" %pool_dhcp)
stdin,stdout,stderr = client.exec_command("ip dhcp-server network add address=%s"
%network_dhcp)
stdin,stdout,stderr = client.exec_command("ip dhcp-server add address-pool=pool1
disabled=no interface=%s name=server1" %bridge_conf)
stdin,stdout,stderr = client.exec_command("radius add address=%s secret=%s
service=wireless", % (radius_ip,radius_secret))
stdin,stdout,stderr = client.exec_command("caps-man security add authentication-types=wpa2-
eap eap-methods=passthrough encryption=aes-ccm group-encryption=aes-ccm
name=security1 tls-certificate=none tls-mode=no-certificates")
stdin,stdout,stderr = client.exec_command("caps-man configuration add country=spain
datapath.bridge=%s datapath.client-to-client-forwarding=yes datapath.local-forwarding=yes
name=cfg1 security=security1 ssid=%s" % (bridge_conf, wifi_ssid))
stdin,stdout,stderr = client.exec_command("caps-man manager interface add disabled=no
interface=%s" % bridge_conf)
```

```
stdin,stdout,stderr = client.exec_command("caps-man provisioning add action=create-dynamic-
enabled master-configuration=cfg1")
stdin,stdout,stderr = client.exec_command("caps-man manager set enabled=yes")
client.close()
```

Configurar OVS1

```
#!/bin/bash
# Borrar configuraciÃ³vs-vsctl del-br sw0
# Crear switch
ovs-vsctl add-br sw0
# Asignar puertos a switch
ovs-vsctl add-port sw0 enp0s3
ovs-vsctl add-port sw0 enp0s8
ovs-vsctl add-port sw0 enp0s9
ovs-vsctl add-port sw0 enp0s10
# Asignar vlan a puerto
ovs-vsctl set port enp0s3 tag=58 vlan_mode=native-untagged
ovs-vsctl set port enp0s8 tag=58 vlan_mode=native-untagged
ovs-vsctl set port enp0s9 tag=58 vlan_mode=native-untagged
ovs-vsctl set port enp0s10 tag=58 vlan_mode=native-untagged
# Crear vlan y asignarla al switch
ovs-vsctl add-port sw0 vlan58 tag=58 -- set interface vlan58 type=internal
# Levantar interfaz y asignar ip
ip link set vlan58 up
ip addr add 192.168.58.240/24 dev vlan58
# Enlazar switch con el controlador
ovs-vsctl set-controller sw0 tcp:192.168.58.253:6633
```

Configurar OVS2

```
#!/bin/bash
# Borrar configuraciÃ³vs-vsctl del-br sw0
# Crear switch
ovs-vsctl add-br sw0
# Asignar puertos a switch
ovs-vsctl add-port sw0 enp0s3
ovs-vsctl add-port sw0 enp0s8
ovs-vsctl add-port sw0 enp0s9
ovs-vsctl add-port sw0 enp0s10
# Asignar vlan a puerto
ovs-vsctl set port enp0s3 tag=58 vlan_mode=native-untagged
ovs-vsctl set port enp0s8 tag=58 vlan_mode=native-untagged
ovs-vsctl set port enp0s9 tag=58 vlan_mode=native-untagged
ovs-vsctl set port enp0s10 tag=58 vlan_mode=native-untagged
# Crear vlan y asignarla al switch
ovs-vsctl add-port sw0 vlan58 tag=58 -- set interface vlan58 type=internal
# Levantar interfaz y asignar ip
ip link set vlan58 up
ip addr add 192.168.58.239/24 dev vlan58
# Enlazar switch con el controlador
ovs-vsctl set-controller sw0 tcp:192.168.58.253:6633
```

Switch Management

```
#!/usr/bin/python3.6
```

```
import os
import requests
import json

def menu():

	os.system('clear')
	print ("Gestion del switch")
	print ("\t1 - Ver tabla de MACs")
	print ("\t2 - Introducir MAC")
	print ("\t3 - Configurar IP Controlador")
	print ("\t4 - Salir\n")

controlador = "127.0.0.1"

while True:
	menu()
	opc = input("Elige una opcion: ")

	if opc=="1":
		sw = input("Introduce el id del switch: ")
		r = requests.get('http://%s:8080/simpleswitch/mactable/%s' % (controlador,sw))
		print(json.dumps(r.json(),indent=1))
		input("Pulsar enter para continuar")
	if opc=="2":
		sw = input("Introduce el id del switch: ")
		sw_mac = input("Introduce la MAC: ")
		sw_port = input("Introduce el puerto: ")
		datos = {}
		datos['mac'] = sw_mac
		datos['port'] = int(sw_port)
		url = "http://" + controlador + ":8080/simpleswitch/mactable/" + sw
		r = requests.put(url, data=json.dumps(datos))
		input("Pulsar enter para continuar")
	if opc=="3":
		controlador = input("Introduce la IP del controlador: ")
	if opc=="4":
		break
	else:
		print("Escoge una opcion valida.\n")
```

Firewall management

```
#!/usr/bin/python3.6

import os
import requests
import json

def menu():

	os.system('clear')
	print ("Gestion del firewall")
	print ("\t1 - Ver estado del Firewall en los switches")
	print ("\t2 - Habilitar Firewall en switch")
	print ("\t3 - Deshabilitar Firewall en switch")
	print ("\t4 - Obtener las reglas de un switch")
	print ("\t5 - Introducir una nueva regla en un switch")
	print ("\t6 - Borrar una regla de un switch")
```

```
        print ("\t7 - Configurar IP del controlador")
        print ("\t8 - Salir\n")

controlador = "127.0.0.1"

while True:
        menu()
        opc = input("Elige una opcion: ")

        if opc=="1":
                r = requests.get('http://%s:8080/firewall/module/status' % controlador)
                print(json.dumps(r.json(),indent=1))
                input("Pulsar enter para continuar")
        if opc=="2":
                sw = input("Introduce el id del switch: ")
                url = "http://" + controlador + ":8080/firewall/module/status/enable" + sw
                r = requests.put(url)
                input("Pulsar enter para continuar")
        if opc=="3":
                sw = input("Introduce el id del switch: ")
                url = "http://" + controlador + ":8080/firewall/module/status/disable" + sw
                r = requests.put(url)
                input("Pulsar enter para continuar")
        if opc=="4":
                sw = input("Introduce el id del switch: ")
                r = requests.get('http://%s:8080/firewall/rules/%s' % (controlador,sw))
                print(json.dumps(r.json(),indent=1))
                input("Pulsar enter para continuar")
        if opc=="5":
                sw = input("Introduce el id del switch: ")
                ip_src = input("Introduce la red origen: ")
                ip_dst = input("Introduce la red destino: ")
                proto = input("Introduce el protocolo: ")
                action = input("Introduce la accion: ")
                prio = input("Introduce la prioridad: ")
                datos = {}
                datos['nw_src'] = ip_src
                datos['nw_dst'] = ip_dst
                datos['actions'] = action
                datos['nw_proto'] = proto
                datos['priority'] = prio
                url = "http://" + controlador + ":8080/firewall/rules/" + sw
                r = requests.post(url, data=json.dumps(datos))
                input("Pulsar enter para continuar")
        if opc=="6":
                sw = input("Introduce el id del switch: ")
                pol = input("Introduce el ID de la regla: ")
                url = "http://" + controlador + ":8080/firewall/rules/" + sw
                datos = {}
                datos['rule_id'] = pol
                r = requests.delete(url, data=json.dumps(datos))
                input("Pulsar enter para continuar")
        if opc=="7":
                controlador = input("Introduce la IP del controlador: ")
        if opc=="8":
                break
        else:
                print("Escoge una opcion valida.\n")
```

8.3 Anexo 3: Utilidades utilizadas

Habilitación de la virtualización anidada en VirtualBox

Esto se ha realizado debido a la necesidad de virtualizar máquinas virtuales dentro de otra máquina virtual.

```
C:\Program Files\Oracle\VirtualBox> VBoxManage.exe modifyvm NFV --nested-hw-virt on
```

Obtención de un fichero qcow2 a partir de una OVA

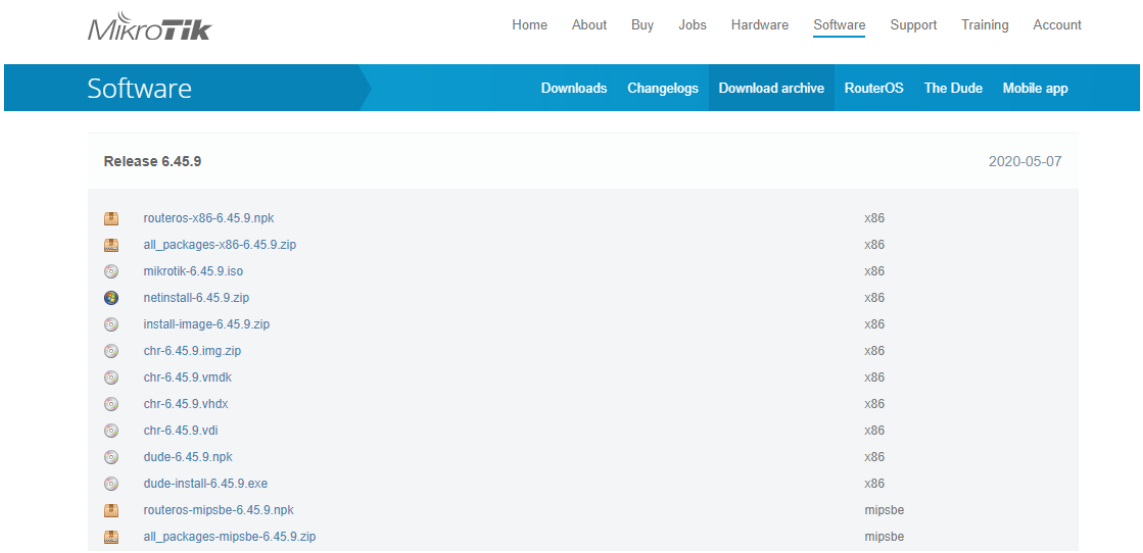
```
# tar xvf maquina.ova
# qemu-img convert -O qcow2 maquina.vmdk maquina.qcow2
```

8.4 Anexo 4: Habilitar OpenFlow en Mikrotik

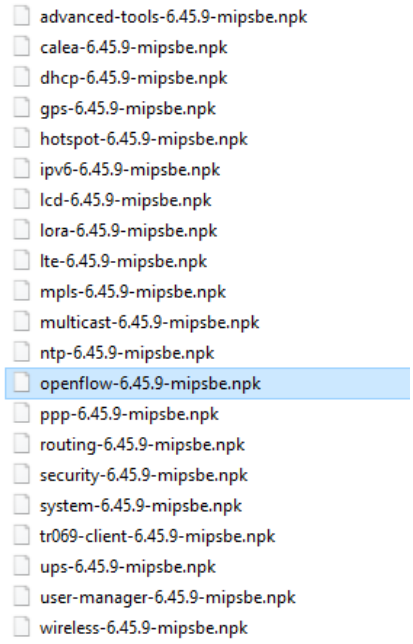
Por defecto Mikrotik no trae habilitada la característica de OpenFlow por lo que hay que habilitarla manualmente. Para realizar esto es necesario descargar los paquetes necesarios y subirlos al equipo.

Es necesario conocer la versión de software RouterOS instalada en el equipo (en este caso 6.45.9) y buscar el paquete para descargar en el siguiente enlace: https://mikrotik.com/download/archive

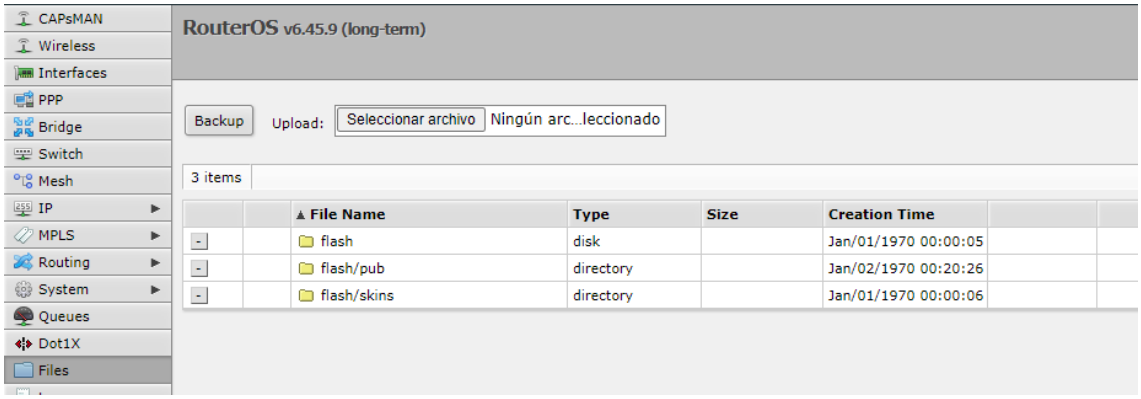
El paquete en cuestión es all_packages-mipsbe-6.45.9.zip



Una vez descargado el paquete se descomprime para ver todas las opciones adicionales que se pueden instalar:



Se escoge el paquete OpenFlow y se sube al equipo Mikrotik desde el menu Files > Seleccionar archivo



Una vez subido el archivo es necesario reiniciar el equipo:

```
[admin@MikroTik] > system reboot
Reboot, yes? [y/N]:
y
system will reboot shortly
```

Al reiniciarse el equipo ya queda habilitado OpenFlow