Análisis de redes definidas por software (SDN) y su aplicación en el entorno sanitario

Carlos Mediero Martí
Máster Universitario en Ingeniería de Telecomunicación
Telemática

Nombre Consultor/a:
José López Vicario

Nombre Profesor/a responsable de la asignatura
Xavi Vilajosana Guillen

03/01/2021

B) GNU Free Documentation License (GNU FDL)

C) Copyright

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Análisis de redes definidas por software (SDN) y su aplicación en el entorno sanitario*
Nombre del autor:	*Carlos Mediero Martí*
Nombre del consultor/a:	*José López Vicario*
Nombre del PRA:	*José López Vicario*
Fecha de entrega (mm/aaaa):	01/2021
Titulación:	Máster Universitario en Ingeniería de Telecomunicación
Área del Trabajo Final:	*Telemática*
Idioma del trabajo:	*Castellano*
Palabras clave	*SDN, Mininet, ONOS*

Resumen del Trabajo:

Dada la actual evolución de las redes donde se va dejando de lado la antigua arquitectura jerárquica de cliente/servidor, en este TFM se plantea el análisis y estudio del nuevo paradigma de las redes definidas por software (SDN) para su implantación en el entorno sanitario y dotar a las redes de los hospitales de una mayor eficiencia, flexibilidad y escalabilidad.

Por ello, en la primera parte del TFM se recopilará información acerca de esta tecnología desde sus inicios hasta la actualidad para poder estudiarla y entender su funcionamiento.

Posteriormente se analizarán las diferentes herramientas que se emplearán para el desarrollo del TFM. Para el diseño de red se optará por la herramienta Mininet con su entorno de trabajo gráfico miniedit y para las funciones de controlador se optará por el controlador ONOS.

Una vez entendida la tecnología, se realizará un estudio de la arquitectura de red actual de un hospital, para ello se tomará como referencia el modelo de red planteado por Ignacio Campos en su TFM [22]. Se hará hincapié en la topología de red implantada, la electrónica de red que la conforma y los distintos entornos y VLAN`s desplegados en el mismo.

Por último, se planteará el diseño de red para la implantación de SDN en la actual topología de red del hospital, se reproducirá el escenario empleando Mininet y el controlador ONOS. Se realizarán simulaciones para monitorización de tráfico en las sedes, se aplicará VPLS y se diseñará un clúster del controlador ONOS para dotar al diseño de alta disponibilidad y redundancia ante fallos.

Abstract:

Given the current evolution of networks, the old hierarchical client / server architecture is being left behind. As a result, this dissertation will analyse and explore/investigate/study the new paradigm of Software-Defined Networking (SDN) for its implementation in the healthcare setting and to provide hospitals with more efficient, flexible and scalable networks.

In the first section of this dissertation, data will be collected regarding the history and development of this new technology in order to fully comprehend how it works.

Thereafter, the different tools used within this study will be analysed. The tool Mininet and its graphical user interface MiniEdit, will be used for the network design and the ONOS controller will be used for the controller functions.

Upon understanding the technology, a study will be carried out on a hospitals' current network architecture of which the network model presented in Ignacio Campos' dissertation [22] will be used as a reference. This study will place specific emphasis on the topology of network installed, the network electronics that it is comprised of as well as the different environments and VLANs deployed in it.

Finally, the network design for the implementation of SDN in the current hospital network topology will be presented, the scenario will be reproduced using Mininet and the ONOS controller and simulations will be carried out to monitor traffic at the headquarters, VPLS will be applied, and a cluster of the ONOS controller will be designed to provide the design with high availability and redundancy in the event of failures.

Índice

Lista de figuras

Lista de tablas

1. Introducción

1.1 Contexto y justificación del Trabajo

Debido a la evolución de las nuevas tecnologías que se ha experimentado en los últimos años como, por ejemplo, la aparición de numerosos dispositivos que se conectan a la red, el tratamiento de grandes volúmenes de datos, streaming, cloud computing, etc. las redes tradicionales presentan limitaciones para adaptarse a las nuevas exigencias de los usuarios y empresas ya que resultan estáticas, poco escalables, difíciles de mantener y complejas en su gestión y administración.

Es por ello por lo que las redes tradicionales están experimentando una evolución hacia el nuevo paradigma de las redes definidas por software (SDN), de esta manera la redes podrán adaptarse a las nuevas exigencias de los usuarios y empresas.

Bajo estas nuevas exigencias con aplicaciones críticas y nuevas tecnologías emergentes, las redes SDN son de gran importancia ya que aportan una solución global y son una alternativa a las redes tradicionales. Estas nuevas redes son capaces de soportar las altas exigencias de los servicios actuales y futuros gracias a su adaptación dinámica a las necesidades de cada servicio, lo que proporciona al mismo tiempo una reducción de los tiempos de implantación de nuevos servicios.

Por consiguiente, en este TFM se plantea el análisis y estudio del nuevo paradigma de las redes definidas por software (SDN) para su implantación en el entorno sanitario y dotar a las redes de los hospitales de una mayor eficiencia, flexibilidad y escalabilidad.

Partiendo del modelo de red planteado por Ignacio Campos Marcé en su proyecto TFM [22] con título “Diseño de red de una organización Sanitaria”, plantearemos su aplicación a las SDN.

Se planteará el diseño de red para la implantación de SDN en la actual topología de red del hospital, para ello se realizarán simulaciones para monitorización de tráfico en las sedes, ya que hoy en día no se dispone de ninguna herramienta que de una manera rápida y visual se identifique un elemento que causa saturación en la red. Se continuará con aplicación de Virtual Private LAN Service (VPLS) en el diseño propuesto para habilitar la comunicación entre dispositivos que pertenecen a la misma VLAN sin importar en qué sede se encuentran ubicados. De esta manera se conseguirá que todos los hospitales parezcan estar en la misma LAN Ethernet y se dotará de mayor seguridad a la red.

Finalmente se añadirá al diseño planteado la implementación de un clúster del controlador ONOS para formar un sistema distribuido

unificado y dotar al diseño propuesto de alta disponibilidad y redundancia ante fallos.

1.2 Objetivos del Trabajo

El principal objetivo de este TFM es el análisis y estudio de las redes definidas por software (SDN) para su implantación en el entorno sanitario y dotar a las redes de los hospitales de una mayor eficiencia, flexibilidad y escalabilidad.

No obstante, se definen los siguientes objetivos para llevar a cabo en este trabajo:

- Estudio del estado actual de las redes definidas por software y entender su funcionamiento para saber aplicarlo.

- Conocimiento de las herramientas y componentes que se emplean en las SDN, como los controladores y los emuladores de red.

- Estudio de la topología de red actual de un hospital y diseño de una red definida por software para su adaptación en el entorno sanitario para dotar a las redes de los hospitales de una mayor eficiencia, flexibilidad y escalabilidad, y facilitar la labor de los administradores de red.

- Implantación de Virtual Private LAN Service (VPLS) en ONOS para habilitar la comunicación entre equipos de la misma VLAN sin importar en qué sede se encuentran ubicados.

- Implementación de un clúster del controlador ONOS para formar un sistema distribuido unificado y dotar al diseño propuesto de alta disponibilidad y redundancia ante fallos.

1.3 Enfoque y método seguido

Debido a los objetivos establecidos en el TFM, se ha optado por enfoque práctico donde se plantearán diversos modelos de red y simulaciones que permitan:

- Monitorización del tráfico en tiempo real, para identificar y controlar equipos que puedan ocasionar saturación en la red.

- Implantación de Virtual Private LAN Service (VPLS) en ONOS para habilitar la comunicación entre equipos de la misma VLAN sin importar en qué sede se encuentran ubicados. De esta manera se conseguirá que todos los hospitales parezcan estar en la misma LAN Ethernet y se le aportará seguridad a la red.

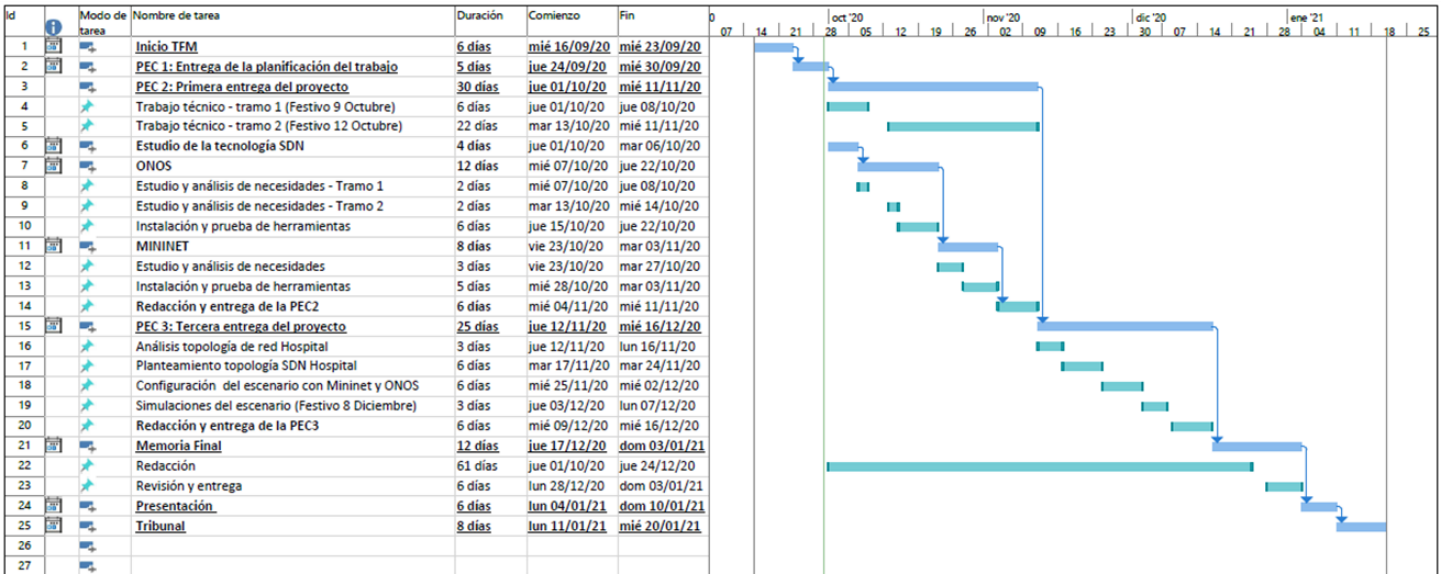
- Implementación de un clúster del controlador ONOS para formar un sistema distribuido unificado y dotar al diseño propuesto de alta disponibilidad y redundancia ante fallos.

1.4 Planificación del Trabajo

Para llevar a cabo este TFM se llevará a cabo la instalación de diversos recursos para la puesta en marcha de los entornos necesarios y la ejecución de los diseños y simulaciones planteados.

Dentro de los recursos necesarios se encuentran los siguientes:

- Software VirtualBox para la ejecución de máquinas virtuales.
- Imagen del S.O. Ubuntu Server 18.04.
- Software JDK de Java.
- Software Mininet para emulación de redes
- Software Mininet-Wifi para emulación de redes inalámbricas.
- Software de controlador ONOS



Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin
1		Inicio TFM	6 días	mié 16/09/20	mié 23/09/20
2		PEC 1: Entrega de la planificación del trabajo	5 días	jue 24/09/20	mié 30/09/20
3		PEC 2: Primera entrega del proyecto	30 días	jue 01/10/20	mié 11/11/20
4		Trabajo técnico - tramo 1 (Festivo 9 Octubre)	6 días	jue 01/10/20	jue 08/10/20
5		Trabajo técnico - tramo 2 (Festivo 12 Octubre)	22 días	mar 13/10/20	mié 11/11/20
6		Estudio de la tecnología SDN	4 días	jue 01/10/20	mar 06/10/20
7		ONOS	12 días	mié 07/10/20	jue 22/10/20
8		Estudio y análisis de necesidades - Tramo 1	2 días	mié 07/10/20	jue 08/10/20
9		Estudio y análisis de necesidades - Tramo 2	2 días	mar 13/10/20	mié 14/10/20
10		Instalación y prueba de herramientas	6 días	jue 15/10/20	jue 22/10/20
11		MININET	8 días	vie 23/10/20	mar 03/11/20
12		Estudio y análisis de necesidades	3 días	vie 23/10/20	mar 27/10/20
13		Instalación y prueba de herramientas	5 días	mié 28/10/20	mar 03/11/20
14		Redacción y entrega de la PEC2	6 días	mié 04/11/20	mié 11/11/20
15		PEC 3: Tercera entrega del proyecto	25 días	jue 12/11/20	mié 16/12/20
16		Análisis topología de red Hospital	3 días	jue 12/11/20	lun 16/11/20
17		Planteamiento topología SDN Hospital	6 días	mar 17/11/20	mar 24/11/20
18		Configuración del escenario con Mininet y ONOS	6 días	mié 25/11/20	mié 02/12/20
19		Simulaciones del escenario (Festivo 8 Diciembre)	3 días	jue 03/12/20	lun 07/12/20
20		Redacción y entrega de la PEC3	6 días	mié 09/12/20	mié 16/12/20
21		Memoria Final	12 días	jue 17/12/20	dom 03/01/21
22		Redacción	61 días	jue 01/10/20	jue 24/12/20
23		Revisión y entrega	6 días	lun 28/12/20	dom 03/01/21
24		Presentación	6 días	lun 04/01/21	dom 10/01/21
25		Tribunal	8 días	lun 11/01/21	mié 20/01/21
26					
27					

Figura 1: Diagrama de Gantt de la planificación del TFM

Para llevar a cabo la planificación se ha tomado como base las fechas de entrega de los distintos trabajos a entregar para la evaluación continua del TFM.

A continuación, se mostrará el seguimiento de la planificación establecida:

PEC1: Entrega de planificación del trabajo:

- Sin anomalías
- Se sigue la planificación del trabajo sin producirse retrasos.

PEC2: Primera entrega del proyecto:

- Tras entrega de la PEC1, es necesaria corrección en la definición de los objetivos, hay que redefinirlos y aplicar métricas
    - Coste temporal: 1 día sobre el proyecto.

- Se recupera coste temporal y se sigue la planificación del trabajo sin producirse más retrasos.

PEC3: Segunda entrega del proyecto:

- Debido a problemas personales se ve gravemente retrasado el avance del proyecto. No se ha conseguido cumplir con la planificación inicial
    - Coste temporal: 6-7 días sobre el proyecto

- Para intentar recuperar el retraso, se aumentará las horas de dedicación diarias en la medida de lo posible.

MEMORIA FINAL:

- Se logra recuperar el retraso acumulado hasta la fecha cumpliendo con planificación inicial.

1.5 Breve sumario de productos obtenidos

Debido al tipo de proyecto propuesto y al método de ejecución, los productos obtenidos en este TFM se basarán en los resultados proporcionados después de los diseños de red y simulaciones propuestas.

Se adjuntarán en los anexos el código de las redes diseñadas con Mininet para las distintas simulaciones.

1.6 Breve descripción de los otros capítulos de la memoria

Además de este capítulo de introducción, donde se dan a conocer los aspectos y puntos más relevantes a tratar en el TFM, se ha estructurado esta memoria en torno a 4 capítulos junto con un capítulo final asociado a las conclusiones del trabajo. A continuación, se realiza una breve descripción de estos capítulos:

Capítulo 2: SDN

En este capítulo, se discutirá el concepto de SDN y las nuevas funciones que trae en relación con la red actual. Se analizará el estado actual de la

tecnología y su arquitectura general, así como el papel relevante del protocolo OpenFlow en la misma. También se revisará las diferentes ventajas que brinda esta tecnología en comparación a las redes tradicionales.

Capítulo 3: ONOS y MININET

En este capítulo se estudian en detalle las dos herramientas principales para la elaboración del TFM.

Se analizará el software del controlador SDN de ONOS y las diferentes funciones que proporciona, además, se revisarán las especificaciones técnicas del mismo.

Se estudiará el funcionamiento del emulador de redes Mininet, sus comandos y principales características.

Capítulo 4: Estudio modelo de red actual de un hospital

En este capítulo se realizará un estudio de la topología de red actual de un hospital para posteriormente, en el capítulo 5, plantear el diseño de una red definida por software para su adaptación en el entorno sanitario.

Para ello nos basaremos en el modelo planteado por Ignacio Campos en su TFM [22].

Capítulo 5: Diseño de red propuesto

En este capítulo se plantea la adaptación del diseño de red actual de los hospitales y las simulaciones correspondientes para su adaptación a las SDN.

Se plantean diseños para:

- Monitorización del tráfico en tiempo real, para identificar y controlar equipos que puedan ocasionar saturación en la red.

- Implantación de Virtual Private LAN Service (VPLS) en ONOS para habilitar la comunicación entre equipos de la misma VLAN sin importar en qué sede se encuentran ubicados. De esta manera se conseguirá que todos los hospitales parezcan estar en la misma LAN Ethernet y se le aportará seguridad a la red.

- Implementación de un clúster del controlador ONOS para formar un sistema distribuido unificado y dotar al diseño propuesto de alta disponibilidad y redundancia ante fallos.

2. SDN

2.1 Descripción

SDN (Software-Defined Networking)[1][5][6][23] es el nuevo paradigma que permite controlar y gestionar las redes de comunicaciones de una manera centralizada gracias al desacoplamiento entre el plano de control y el plano de datos. En el plano de datos es donde se localizan los puertos de los elementos de red y donde se realiza el tratamiento de los datos, reenviándolos por las diferentes interfaces de los elementos de red según las reglas y lógicas aplicadas en el plano de control.

En las redes tradicionales nos encontramos estos planos integrados en los elementos de red (switches/routers), por lo tanto, son estos elementos los encargados de llevar a cabo las funcionalidades deseadas en la red. Esto es un hándicap, ya que para adaptar la red a una nueva funcionalidad se hace necesaria la configuración y gestión manual de cada uno de los elementos que conforman la misma. Por lo tanto, la administración de grandes redes se convierte en una tarea tediosa.

Como hemos mencionado anteriormente, con la implantación de SDN, además de separar los planos y centralizar la gestión de la red, se abre a la posibilidad de realizar dicha gestión a través de software con una consola de control centralizada conocida como SDN Controller, de esta manera lograremos que el despliegue de nuevas redes o la implementación de nuevas funcionalidades y configuraciones se puedan realizar de una manera más ágil y flexible.

2.2 Estado del arte

El aumento de usuarios, dispositivos y aplicaciones derivados del Internet de las cosas (IoT), del proceso y análisis de grandes cantidades de datos (Big Data), o plataformas de Streaming, han provocado el crecimiento de las redes y su vez, la demanda de los sistemas ante la necesidad de mayores recursos de almacenamiento, ancho de banda y alta flexibilidad para que la experiencia del usuario no se vea mermada.

La tendencia a convertir la red de transporte en redes multiservicio, hacen de las SDN el modelo de red idóneo para llevar a cabo esta transformación. El modelo de red tradicional, tratándose de un modelo distribuido presenta diversas dificultades como su limitación a la escalabilidad y su limitación en la operatividad, sin embargo, las SDN con su modelo centralizado y el hecho de tener desagregados la parte de control y la parte del procesado de transporte permite que con sólo un software de control se pueda gestionar remotamente infinidad de equipos. De esta manera obtendremos una red 100% escalable y completamente integrada.

Por lo anterior, han emergido paradigmas para la gestión centralizada de todos los componentes de una red mediante plataformas tecnológicas totalmente administrables, centralizadas y dinámicas como, por ejemplo, SD-WAN (Software Defined-Wide Area Network).

No obstante, teniendo en cuenta los objetivos planteados para el trabajo, para adaptar la red de un hospital al modelo SDN se partirá del modelo planteado por Ignacio Campos Marcé en su TFM [22], pudiendo considerarse el presente trabajo como una continuación o una línea de trabajo futura del TFM de Ignacio Campos.

2.3 Arquitectura

Para observar las mejoras que ofrece el modelo SDN, primero analizaremos las redes tradicionales.

Como ya se ha comentado previamente el modelo de red tradicional [23] se basa en un modelo distribuido compuesto por elementos de red como routers y switches que pueden estructurarse en varias capas:

- **Capa de aplicación:** En esta capa es donde se aplican las funcionalidades de los dispositivos. Por ejemplo, en un router se puede aplicar funcionalidades de NAT (Network Address Translation) o IP routing (estático o dinámico).

- **Capa de control:** es donde se toman las decisiones de encaminamiento entre puertos y priorización de paquetes IP en función da funcionalidad diseñada en la capa de aplicación. Podríamos decir qué sería el sistema operativo del dispositivo. Aquí encontramos el inconveniente de que cada dispositivo posee su software propio fijado por el fabricante.

- **Capa de datos:** en esta capa es donde básicamente se transfieren los paquetes IP entrantes o salientes del router o las tramas Ethernet entre puertos entrantes o salientes de un switch. Esta transferencia dependerá de las decisiones tomadas en la capa de control.

Todas estas capas se integran en el mismo dispositivo hardware y los encaminamientos se deciden de manera local.

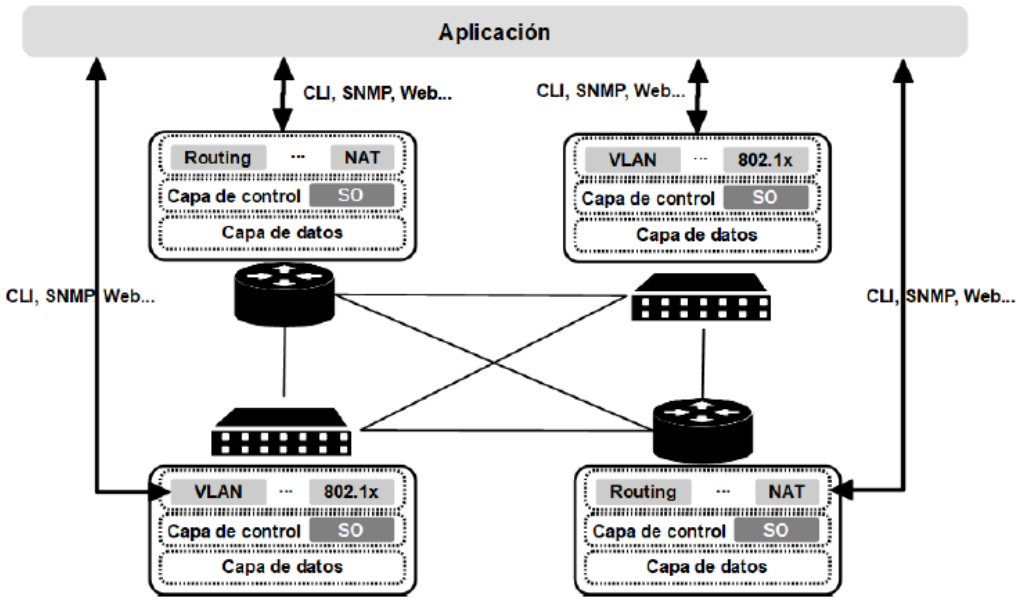


Figura 2: Modelo de red tradicional [23]

No obstante, este modelo de red tradiconal presenta desventajas para su adaptación a las actuales necesidades (IoT, Big Data, Streaming, 5G, etc), incumpliendo los requisitos de robustez, ancho de banda y QoS. Encontramos desventajas como:

- **Rigidez y falta de flexibilidad**, debido a la imposibilidad de configurar varios elementos de red simultáneamente, hace que el proceso de adaptación de dicha red a una aplicación en concreto sea un proceso tedioso, ya que, en este caso, se debe comunicar Individualmente con cada dispositivo y configurarlo 1 a 1.

- **Alto coste de operatividad** derivado de la anterior desventaja, ya que el tiempo dedicado por ingenieros para esa adptación sería muy elevado.

2.4 Capas modelo SDN

Como ya hemos mencionado anteriormente, SDN introduce un nuevo paradigma donde se busca el desacoplamiento entre la capa de datos y la capa de control. Para ello se propone un modelo totalmente centralizado donde un solo un elemento concentra la inteligencia y el conocimiento de toda la red (**SDN Controller)** y permite el control remoto de todos los elementos de red que conforman la capa de infraestructura y que procesan el tráfico o los paquetes.

El SDN Controller es el principal elemento de un entorno SDN y es el responsable de comunicar información a la Capa de infraestructura, formada por switches o routers programables, a través de la interfaz Southbound, y a la capa de aplicación, formada por aplicaciones que se ejecutan en máquinas físicas y virtuales, a través de la interfaz Northbound

En la siguiente imagen podéis observar el modelo de arquitectura de las SDN

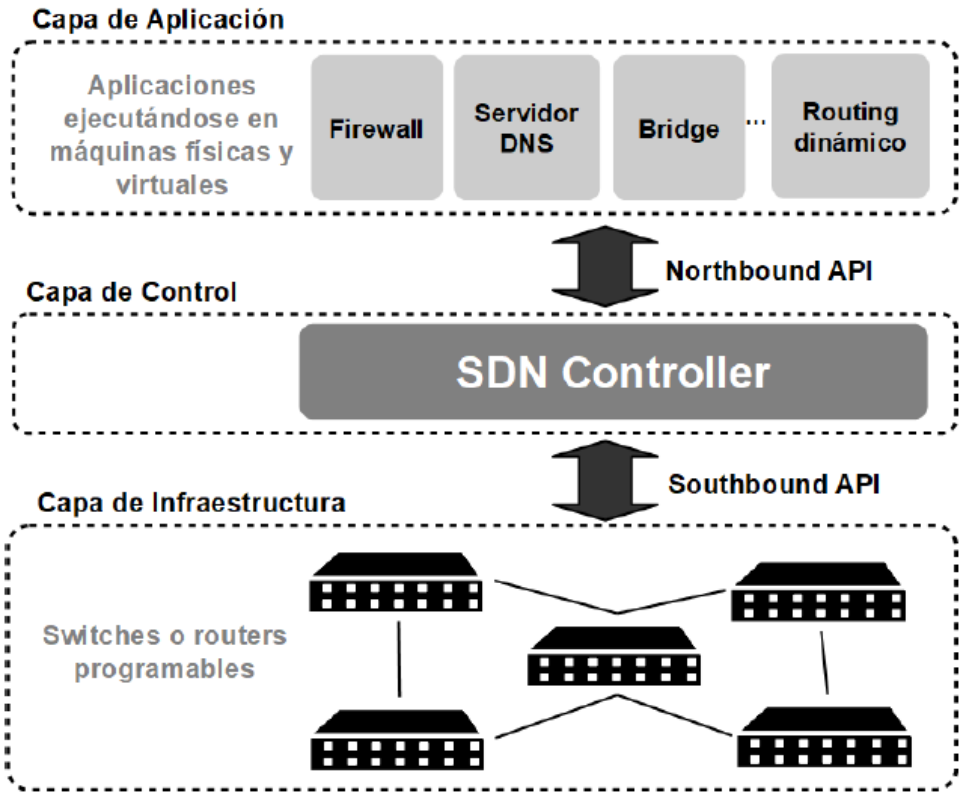


Figura 3: Modelo de arquitectura de las SDN [23]

En este modelo encontramos las siguientes capas [23]:

- **Capa de aplicación:** esta capa aloja todas las aplicaciones que se ejecutan en máquinas físicas y virtuales. Se comunican con la capa de control a través de protocolos **Northbound API** como, por ejemplo, Openstack, REST y CloudStack. Estos APIS son proporcionadas por el SDN Controller y los desarrolladores de aplicaciones deben tenerlos en cuenta a la hora de desarrolar sus aplicaciones.

- **Capa de control:** esta capa es la de mayor importancia ya que alberga el controlador (**SDN Controller**). Como ya hemos mencionad, el SDN Controller es un elemento que concentra la inteligencia y el conocimiento de toda la red, por ello, es el encargado de la toma de decisiones sobre el encaminamiento de los datos dentro de esta.

- **Capa de infraestructura:** esta capa alberga todos los elementos de red programables que conmutan el tráfico de la red (switches o routers), estos deben estar habilitados para SDN, y se comunican con la capa de control a través de protocolos **Southbound API,** como OpenFlow, NETCONF y OVSDB.

En resumen, las aplicaciones informarán al SDN Controller de las necesidades de uso de la red para que funcione la aplicación, el SDN Controller lo procesará y, posteriormente, comunicará los cambios de configuración necesarios en los elementos de red (switches o routers) para un correcto funcionamiento de esta.

2.5 SDN Controller

El SDN Controller es el punto estratégico de control dentro de un entorno SDN. Se puede considerar como un sistema operativo de red sobre el que se desarrollan multitud de aplicaciones y el cuál está compuesto por diversos programas que están estructurados en forma de módulos y plugins. Estos plugins dotan al controlador de diferentes funcionalidades que pueden ser aplicadas sobre la red, además, al mismo tiempo pueden ser empleadas por las aplicaciones a modo de 'servicios'.

Existen numerosas opciones de controladores dónde elegir, tanto de código libre cómo propietarios. Hay que tener en cuenta que la elección del SDN Controller vendrá condicionada por la finalidad de la red a gestionar.

Hay numerosas las opciones open source, entre otras, encontramos:

- **NOX/POX:** Es un controlador [24] de código abierto Openflow cuyo propósito es proporcionar una plataforma simplificada para escribir software de control de red en C++ o Python. Fue desarrollado por 'nicira'.

- **Beacon:** es un controlador [25] OpenFlow rápido, multiplataforma, modular y basado en Java que admite operaciones tanto basadas en eventos como con subprocesos

- **OpenDaylight (ODL):** es el proyecto [26] que lidera la implementación de SDN con plataformas de desarrollo libres, razón por la cual muchos miembros de la industria lo consideran un estándar predeterminado. Se trata de la plataforma de controlador SDN de código abierto más implementada que permite personalizar y automatizar redes de cualquier tamaño y escala. Esto es debido a que incluye todas las funcionalidades y aspectos requeridos en un entorno SDN, como el software del controlador y los protocolos de comunicación.

- **ONOS:** es un proyecto [9] Open-Source que se incorporó como proyecto colaborativo a The Linux Foundation en 2014. Su principal objetivo es Proporcionar a los proveedores de servicios de comunicaciones un controlador SDN con alta disponibilidad, escalabilidad y rendimiento. Más adelante, lo evaluaremos más en detalle. Admite protocolos Southbound como como: BGP, OpenFlow y OVSDB.

- **WMware NSX:** Es una plataforma de virtualización [27] con la que se pretendía ampliar las tecnologías relacionadas con la virtualización de redes. Aparece en 2012, justo un año después de la compra de Nicira por parte de VMware, y su uso es popular en soluciones integrales para Cloud Computing y SD-WAN ya que permite optimizar las operaciones en la nube pública y privada. Además, ofrece características de balanceo de carga y firewall.

- **Huawei Agile Controller:** este controlador [28] cubre un amplio abanico de mercados (LAN, WAN, IoT, DCN) al ofrecer una implementación automática, control inteligente y ajustes de ancho de banda bajo demanda, mejorando así la optimización de recursos en la nube y la experiencia del usuario final. Este controlador es adaptable a los ecosistemas ONOS, ODL y los protocolos soportados por su plataforma SDN son: OpenFlow, BGP, IGP, OVSDB y CLI. Sus principales códigos de programación están escritos en Python y JAVA.

Ante la necesidad de escalabilidad, de capacidad de gestión, redundancia o separación de redes, es posible emplear más de un SDN Controller. No obstante, estos SDN Controllers deben estar coordinados y para ello existe la interfaz horizontal SDN (East-West interface) que es gestionada por ellos mismos.

A través de la SDN, los controladores se intercambiarán información como topología de red, eventos, requerimientos de QoS para aplicaciones, etc.

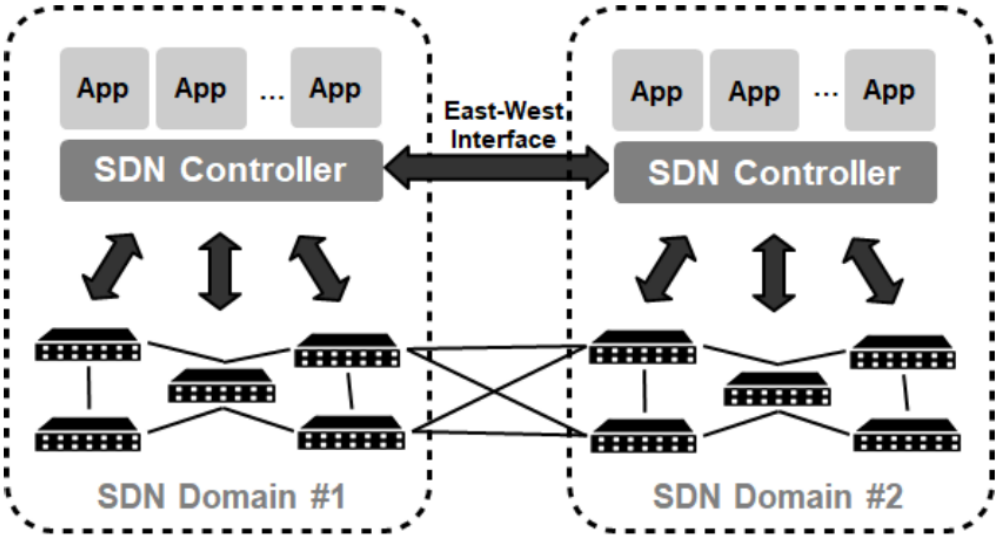


Figura 4: Interconexión entre dos SDN Controllers [23]

2.6 Protocolos

2.6.1 Southbound APIs

Estas son las que hacen posible el desacoplo entre la capa de control y la capa de datos. Existen multitud de APIs en forma de protocolos que los controladores emplean para programar los elementos de red como switches o routers. OpenFlow es el protocolo que se considera como estándar.

2.6.1.1 Openflow

Este protocolo [3][6] surgió a raíz del proyecto de Investigación: "OpenFlow: Enabling Innovation in Campus Networks" de 2008 en la Universidad de Stanford. Actualmente el desarrollo de OpenFlow es liderado por **Open Networking Foundation (ONF),** que se trata de una organización formada por usuarios que concentran su esfuerzo en avanzar en el desarrollo de estándares abiertos y la adopción de tecnologías SDN.

Gracias a este protocolo el controlador puede manipular las tablas de flujo de los elementos de red agregando, actualizando y eliminando flujos de entrada. Esta comunicación se realiza a través de un canal seguro.

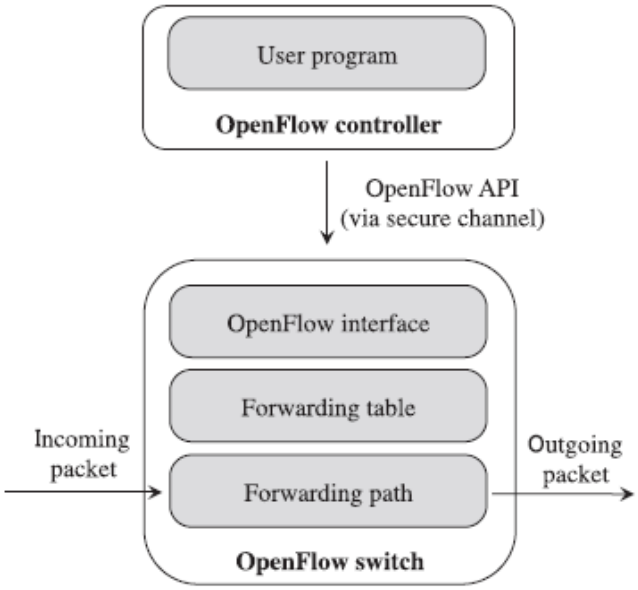


Figura 5: OpenFlow [3]

Las tablas de flujo se emplean para determinar cómo deben ser tratados los paquetes cuando son recibidos, sus entradas contienen un conjunto de campos de los paquetes con los que puede coincidir como, por ejemplo, la dirección ip, y un conjunto de acciones a realizar en función del campo que haya coincidido como, por ejemplo, enviar paquete por el puerto 'x', modificar campo, descartar paquete, etc.

Figura 6: Componentes de un flujo de entrada en una tabla de Flujo Openflow [3]

Una tabla de flujo consta de entradas de flujo que contienen los siguientes campos:

- **match fields:** para emparejar paquetes. Estos consisten en puertos de entrada y encabezados de paquetes, Y otros campos de canalización opcionales, como los metadatos especificados en la tabla anterior.
- **priority:** para marcar la prioridad de la entrada de flujo.
- **counters:** se incrementará cuando el paquete coincida.
- **instructions:** indicara la acción a realizar.
- **timeouts:** marca el tiempo máximo o el tiempo de inactividad antes de que el switch descarte el tráfico.
- **cookie:** se trata del valor de datos opaco seleccionado por el controlador. El controlador puede usarlo para filtrar entradas de flujo afectadas por estadísticas de flujo, para modificaciones de flujo y eliminación de flujos.
- **flags:** flags cambia el modo de gestión de la entrada de flujo, por ejemplo, la bandera OFPFF_SEND_FLOW_REM activa el mensaje de eliminación de flujo de la entrada de flujo.

Las entradas de la tabla de flujo se identifican por su campo *match fields* y *priority,* estos dos campos permiten identificar una entrada de flujo única en una tabla de flujo específica.

Se llama flujo de entrada table-miss al flujo de entrada donde se omiten todos los campos y la prioridad es igual a 0. Cuando el switch recibe un paquete que no coincide con ninguna entrada de la tabla, la acción a tomar dependerá de los flujos de entrada de la table-miss. Ese paquete se podrá enviar al controlador, pasarlo a la siguiente tabla de flujos o descartarlo.

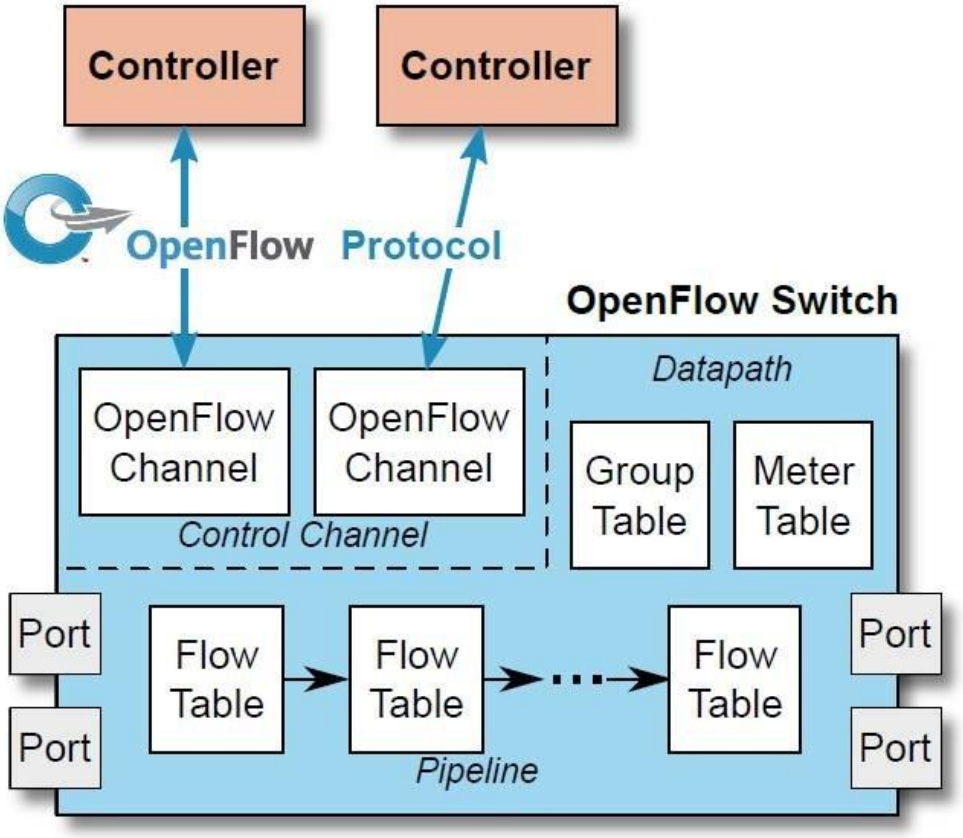


Figura 7: OpenFlow Switch [12]

Si se envía al controlador, esté podrá descartar el paquete o añadir un nuevo flujo de entrada para especificar el siguiente salto y cómo actuar ante paquetes similares en el futuro.

A continuación, se detallará el diagrama de flujo simplificado que detalla el flujo de paquetes a través del OpenFlow Switch [3] y que se muestra en la figura 5.

Al recibir un paquete, un OpenFlow Switch realiza las siguientes operaciones:

- El switch realizará una búsqueda de tabla en la primera tabla de flujo, y dependiendo del pipeline processing, puede llegar a realizar búsquedas de tablas en otras tablas de flujo.
- Los campos *Packet Header* son extraídos del paquete y se recuperan los campos *Packet pipeline*. El campo *Packet Header* es utilizado para realizar la búsqueda en las tablas de flujo, y dependiendo del tipo de paquete de datos se puede realizar la búsqueda mediante la dirección origen Ethernet source address, la dirección ip origen u otro parámetro. Además, el *Packet Header* también se puede comparar con el *ingress port*, el campo de

metadatos y otros campos *pipeline.* Los metadatos se pueden utilizar para pasar información entre tablas en un switch.

- Un paquete coincidirá con un flujo de entrada de la tabla si todos los *match fields* del flujo de entrada coinciden con los campos *Header* y *Pipeline* del paquete.
- Si se omite (valor ANY) un *match field* en el flujo de entrada, este coincidirá con todos los valores posibles en el campo *Header* o *Pipeline* del paquete.
- Si el *match field* está presente y no incluye una máscara, el *match field* coincidirá con el campo *Header* o *Pipeline* del paquete si tiene el mismo valor. Estas búsquedas pueden ser más precisas si el switch permite arbitrariedad de bits en la máscara, donde el *match field* coincidirá con el campo *Header* o *Pipeline* del paquete si tiene el mismo valor para los bits indicados en la máscara.
- El paquete se compara con los flujos de entrada de la tabla de flujos y se selecciona el flujo de entrada con mayor prioridad que coincida con el paquete.
- Los contadores asociados al flujo de entrada seleccionado se actualizan
- Se ejecutan el conjunto de instrucciones incluidos en el flujo de entrada seleccionado.
- Si coinciden varios flujos de entrada y estos tienen la misma prioridad, se considera que el flujo de entrada está explícitamente indefinido.

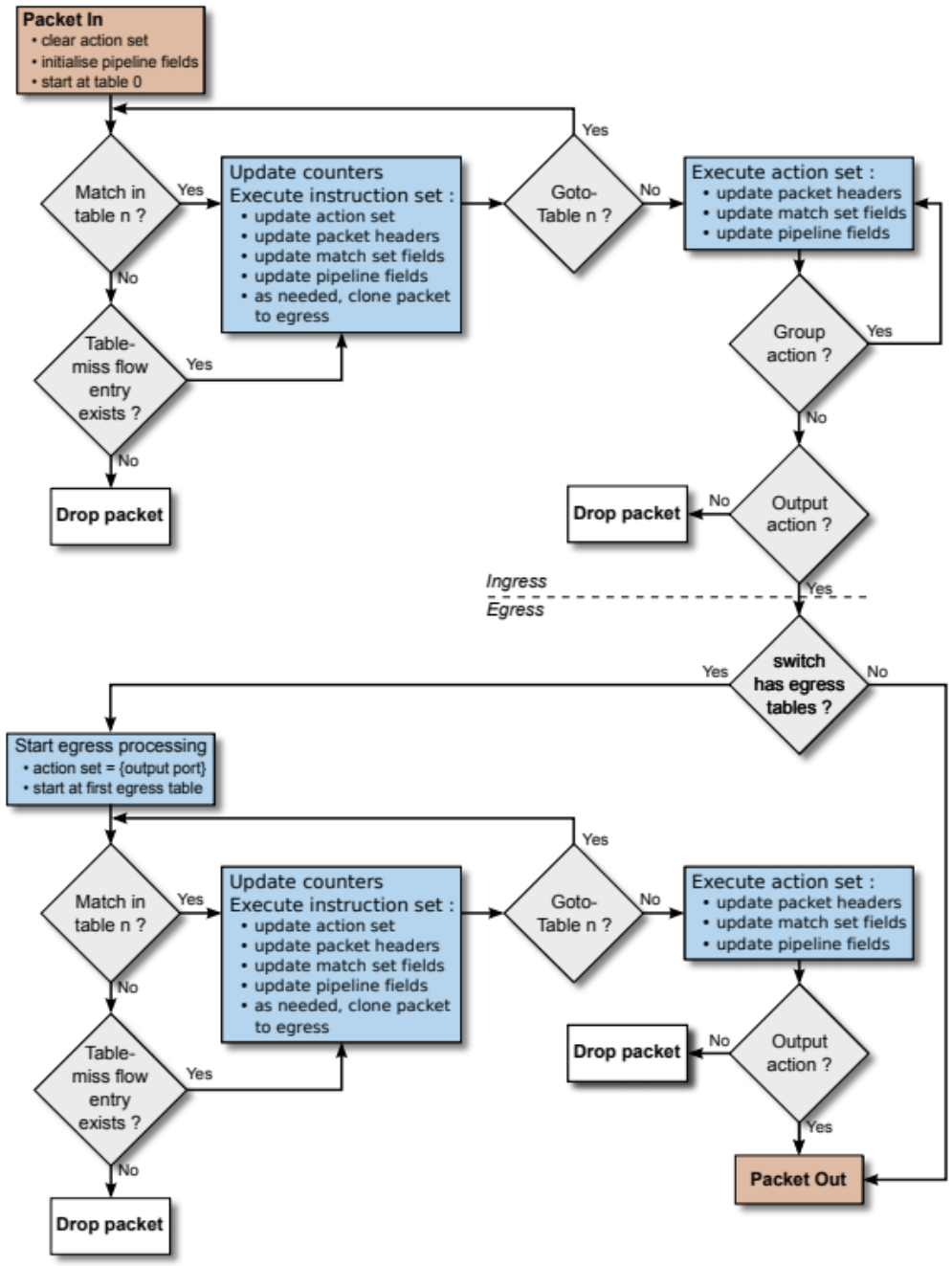


Figura 8: Diagrama de flujo del flujo de paquetes a través del OpenFlow Switch[3]

2.6.1.2 Open vSwitch (OVS) y Open vSwitch Database (OVSDB)

El protocolo OVSDB [7] fue creado por Nicira como protocolo de gestión de Open vSwitch (OVS) [8], que se trata de un software de código abierto diseñado para utilizarse como vswitch (switch virtual) en entornos de servidores virtualizados. Un switch virtual reenvía el tráfico entre diferentes máquinas virtuales (VMs) en el mismo host físico y también reenvía el tráfico entre las VMs y la red física.

Open vSwitch permite su control mediante OpenFlow, para configurar las tablas de flujos que definen las reglas de traspaso y envío de paquetes entre puertos, y el protocolo de gestión OVSDB para para configurar el OVS en sí, pudiendo crear, borrar y modificar puertos e interfaces, así como bridges.

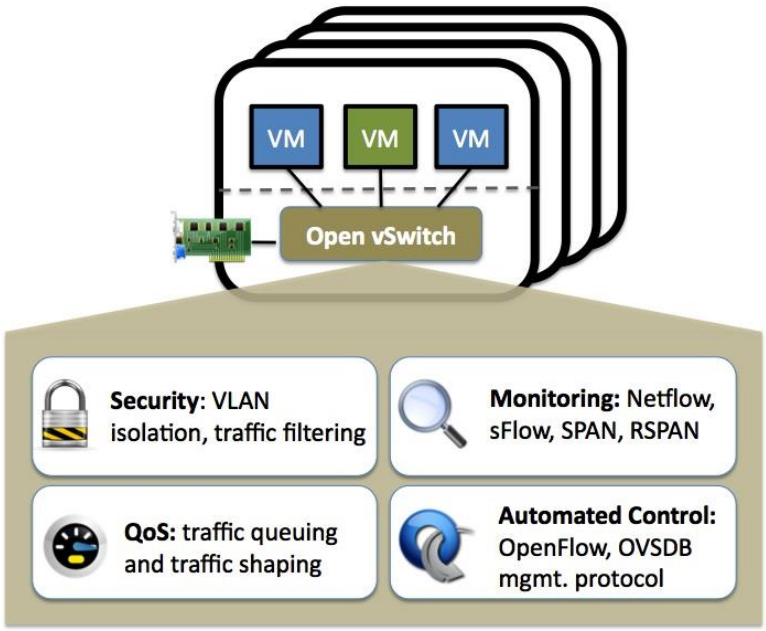


Figura 9: Funcionalidades Open vSwitch [8]

2.6.1.3 Network Configuration Protocol (NETCONF)

Es un protocolo [4] con el que se puede instalar, manipular y eliminar configuraciones de dispositivos de red. Emplea YANG para estructurar la información, XML para codificar los datos de configuración y los mensajes del protocolo, y RPC (remote procedure call) para ejecutar código en una máquina remota.

Cuando un cliente desea editar o consultar la configuración de un elemento de red, codifica un RPC en XML y lo envía a un servidor mediante una conexión segura usando encapsulados como SSH o TLS, este servidor generará una respuesta codificada en XML.

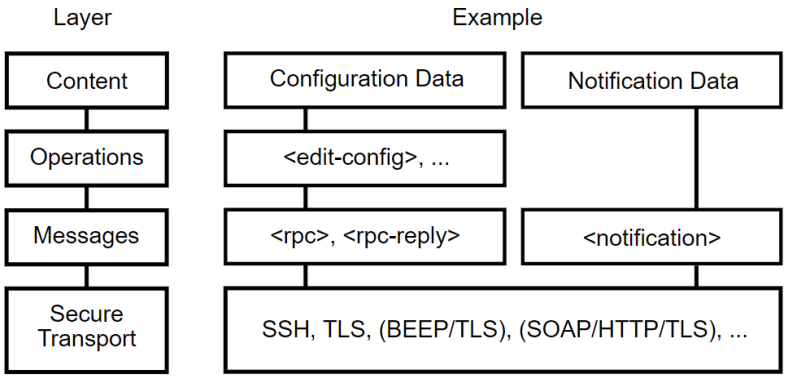


Figura 10: Capas de protocolo NETCONF [4]

2.6.2 Northbound APIs

Como ya hemos comentado anteriormente, el principal objetivo de las *Northbound APIs* es el de permitir a los desarrolladores de aplicaciones realizar las modificaciones pertinentes en la red para un correcto funcionamiento de sus aplicaciones sin la necesidad de conocer la topología ni los dispositivos que la conforman.

Estas APIS son el componente menos estandarizado del entorno SDN ya que cada *Northbound API* da soporte a un tipo de aplicación en concreto. No obstante, la ONF está estudiando la viabilidad de desarrollar un estándar en este componente.

El protocolo más utilizado para implementar las Northbound APIs son los protocolos REST (Representational State Transfer).

REST (Representational State Transfer) [29]: es un protocolo donde los componentes se comunican a través de una interfaz estándar (HTTP) para obtener datos o indicar la ejecución de operaciones sobre los datos en formato XML o JSON. Donde se emplean comandos básicos de HTTP, como son:

- **Post:** Crea nuevos recursos.
- **Get:** Obtiene el recurso solicitado.
- **Put:** Realiza modificaciones.
- **Patch:** Realiza modificaciones de un recurso
- **Delete:** Borra un recurso

2.7 Ventajas SDN

Ya se han podido ir viendo las ventajas de las redes SDN respecto a las redes tradicionales, no obstante, a continuación, las enumeraremos:

Flexibilidad y agilidad

La implantación de redes SDN permite disponer de una infraestructura de red virtualizada que se adapte de manera más rápida a las necesidades de negocio. La red podrá adaptarse según las necesidades de las aplicaciones sin la necesidad de que estas conozcan la topología de red, aportando así una mayor flexibilidad y agilidad.

Gestión centralizada y simplificada

Como ya hemos mencionado, la separación de los planos de datos y control en las redes SDN permite gestionar la totalidad de la red desde un único punto centralizado, eliminando de esta manera el tener que realizar configuraciones individuales en cada uno de los elementos que conforman la red. Es decir, se simplifica la gestión de la red.

Virtualización de la red

Esta se puede realizar mediante NFV (Network Function Virtualization), que es una tecnología que permite que los dispositivos o equipos físicos actuales (firewalls, switches,etc), sean virtualizados mediante el uso de software que se ejecutan en máquinas virtuales.

Programabilidad y automatización

Las SDN facilitan la automatización y adaptación de la red a nuevas funcionalidades o aplicaciones donde las organizaciones pueden liberar a los recursos de realizar tareas tediosas de configuración de red.

Seguridad y privacidad

SDN mejora la seguridad y la privacidad del tráfico de la red gracias a su control de flujo.

Eficiencia y reducción de gastos

Proporciona una mejor utilización de los recursos lo que se deriva en reducción ante la Necesidad de invertir en nuevos bienes físicos y, por consiguiente, en la reducción en gastos de implementación (CAPEX).

Por otro lado, debido a su configuración y gestión de los elementos de red mediante software que aumentan la sencillez y automatización, permite una reducción del tiempo empleado por los administradores de la red y, por consiguiente, la reducción en gastos operación (OPEX).

3. ONOS y Mininet

3.1 ONOS

El Controlador SDN gestiona una extensa cantidad de protocolos de comunicación y distintas capas del modelo OSI. Es necesario conocer cómo funciona ONOS para aplicar sus funcionalidades para conseguir nuestro propósito.

En este capítulo se realizará un estudio de la plataforma de ONOS y las capacidades que puede ofrecer en un entorno SDN.

3.1.1 Arquitectura ONOS

ONOS [9] es un proyecto modular basado en Open Services Gateway initiative (OSGi) y desarrollado en Java. Desde sus inicios, ONOS se ha desarrollado teniendo en cuenta los siguientes aspectos:

- ***Modularidad del código:*** El proyecto consta de diferentes componentes, cada uno de los cuales es un proyecto independiente. Para ello, utiliza las funciones proporcionadas por Maven, que es una herramienta de gestión y comprensión de proyectos de software.

- ***Configurabilidad*****:** Es posible activar o desactivar distintas características tanto en el arranque como en tiempo de ejecución.

- ***Separación de intereses:*** ONOS está dividido en:

    - Módulos orientados a la red con reconocimiento de protocolos que interactúan con la red
    - Núcleo del sistema puede rastrear y proporcionar información sobre el estado de la red. Este núcleo es independiente del protocolo.
    - Aplicaciones que consumen y actúan sobre la información proporcionada por el núcleo

- ***Protocolo de agnosticismo****:* Si ONOS necesita admitir un nuevo protocolo, es posible construir un nuevo módulo orientado a la red contra la Southbound API como un complemento que se puede cargar en el sistema.

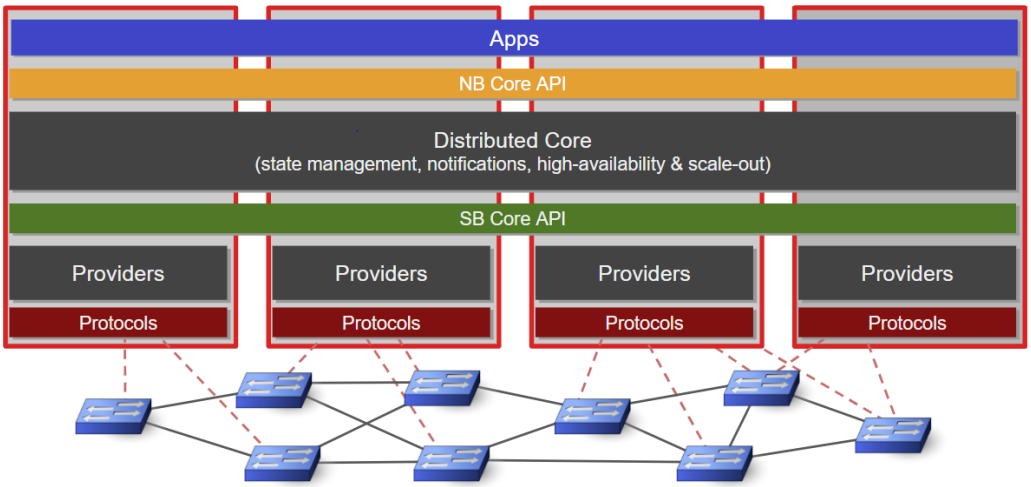


Figura 11: Arquitectura ONOS [9]

3.1.2 Componentes de ONOS

La estructura ONOS consta de uno o más componentes que definen una unidad de funcionamiento dentro del paradigma de ONOS. Este grupo de componentes se conoce como subsistemas. En la siguiente figura se muestran los subsistemas que existen en ONOS:

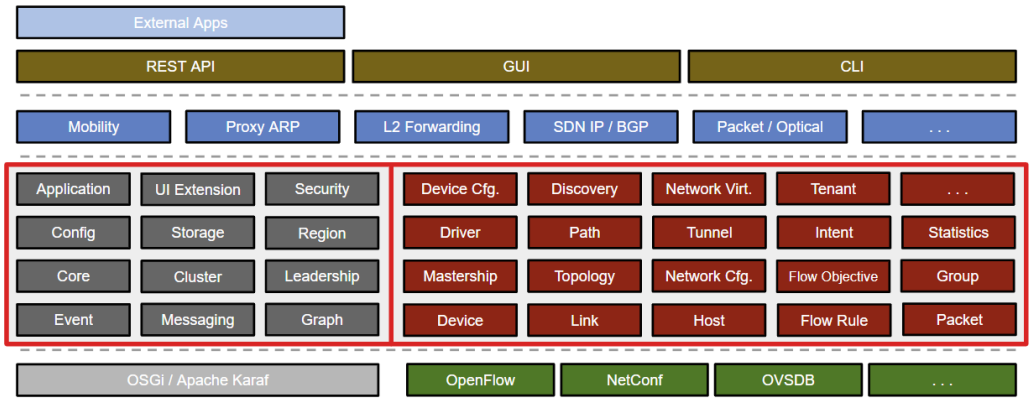


Figura 12: Lista de componentes de ONOS [9]

Existen muchos subsistemas dentro del núcleo de ONOS, pero haremos hincapié en los siguientes:

- ***Device Subsystem:*** Administra el conjunto de dispositivos en la infraestructura.
- ***Host Subsystem:*** Maneja los dispositivos físicos y sus ubicaciones en la red.
- ***FlowRule Subsystem*:** Administra el inventario de las reglas y acciones de flujos en los dispositivos y provee interfaces para estadísticas.

- ***Link Subsystem*:** Gestiona y representa los enlaces físicos entre los dispositivos de red.
- ***Packet Subsystem*:** Administra la comunicación a nivel de paquete de los dispositivos.

3.1.3 Estructura componentes ONOS

Todos los subsistemas pertenecen a uno de los 3 niveles que hay definidos en el paradigma de ONOS: ***App component***, ***Manager Component y Provider component***. En la siguiente figura se muestra la relación existente entre las distintas partes de un subsistema, siendo las marcas de puntos la delimitación de Northbound API y Southbound API.

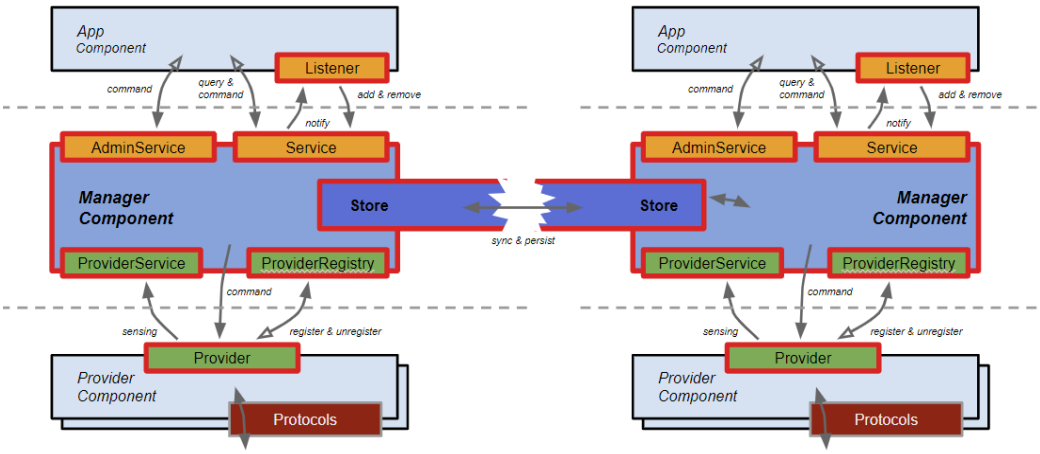


Figura 13: Estructura componentes ONOS [9]

Los componentes se caracterizan por:

- ***Provider Component:***

  Se trata de la capa más baja de ONOS, donde los proveedores interactúan directamente con la red mediante librerías de protocolo específicas y con el núcleo mediante la interfaz *ProviderService.*

  Es posible que algunos proveedores también necesiten admitir comandos de control del núcleo y con el uso de los métodos apropiados del protocolo específico pueden aplicarlos a la red. Estos se trasladan al proveedor a través de la interfaz *Provider*.

- ***Manager Component:***

  Se trata de un componente que se localiza en el núcleo, donde el Manager recibe información de los Providers y lo presenta en la capa de Aplicación y otros servicios.

Se definen las siguientes interfaces:

- La interfaz *AdminService* recibe ordenes de administración y las aplica en la red o el sistema.
- *Service* es la interfaz de Northbound con la que las aplicaciones pueden obtener información sobre el estado de la red.
- *ProviderService* es la interfaz de Southbound mantiene la comunicación e intercambia datos con la capa ***Provider.***
- *ProviderRegistry* es la interfaz de Southbound que permite a los diferentes componentes de la capa ***Provider*** se registren e interactúen con el Manager.

- ***Store***

  Dentro del núcleo y muy relacionado con el Manager, se encuentra ***Store*** que se encargada de indexar, persistir y sincronizar la información que recibe el Manager. Además, debe garantizar la coherencia y solidez de la información en varias instancias de ONOS comunicándose directamente con otros ***stores*** de otras instancias de ONOS.

- ***App Component:***

  Las aplicaciones hacen uso y operan con la información suministrada por los Managers a través de las interfaces *Admin-Service y Service*. Las aplicaciones tienen una amplia gama de funcionalidades como, por ejemplo, mostrar la topología de la red en un navegador web hasta configurar la ruta del tráfico de la red.

3.1.4 Eventos y descripciones

Se trata de las dos unidades básicas de distribución de información en ONOS. Al igual que los servicios, los eventos y las descripciones están asociados con elementos y conceptos específicos de la red.

- **Descripciones**

  Se emplean para proporcionar información de un elemento de la red a través de la southbound API como, por ejemplo, la MAC o la dirección ip de un host.

- **Eventos**

  Son usados por los Managers para notificar de cambios en la topología en la red y por los Stores para notificar los eventos en un entorno distribuido a otros Stores de distintas instancias de ONOS.

Se pueden notificar numerosos eventos como, por ejemplo, que un nuevo dispositivo se ha conectado, que se ha perdido la conexión con un dispositivo o que se ha actualizado un dispositivo.

Cabe destacar que estos eventos sólo llegarán a los oyentes interesados y para ello existen las interfaces *Listener* que se encargarán de capturar los eventos.

La siguiente figura representa la relación entre Descripciones, Eventos y los componentes.

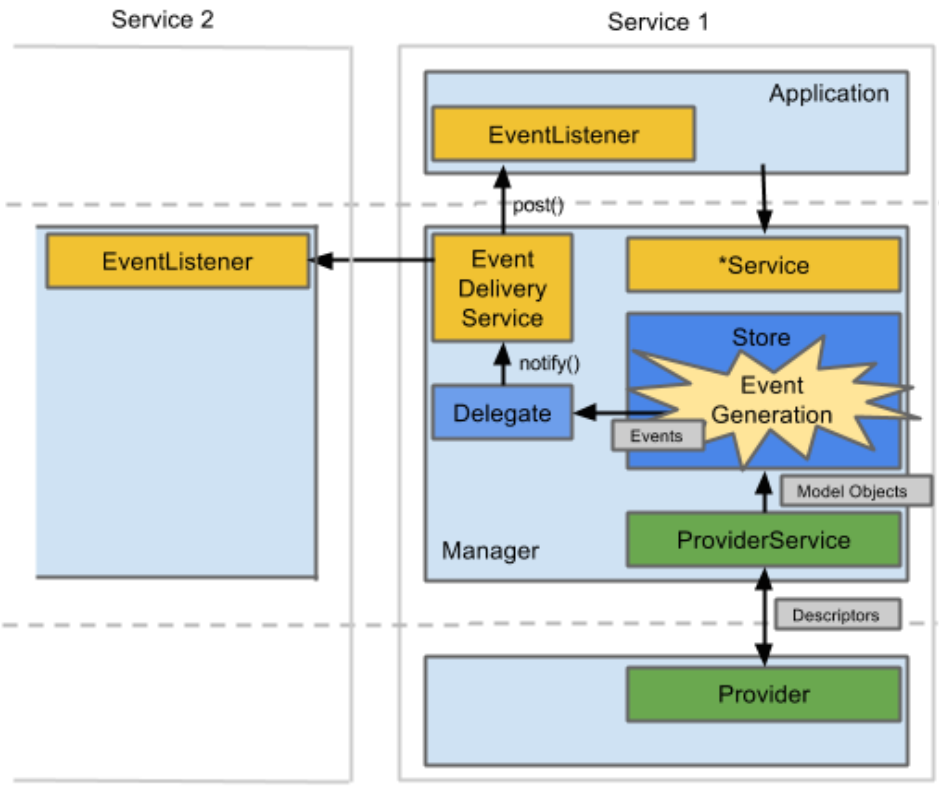


Figura 14: Relación entre Descripciones, Eventos y los componentes [9]

3.1.5 Gestión de ONOS

Se ofrece a los usuarios la posibilidad de interactuar con ONOS a través de tres interfaces:

- **CLI (Línea de comandos):** suele ser la interfaz administrativa principal para una instancia de ONOS. Es capaz de aprovechar características como la extensibilidad programática, la capacidad de cargar y descargar paquetes (entre otros) y el acceso SSH.

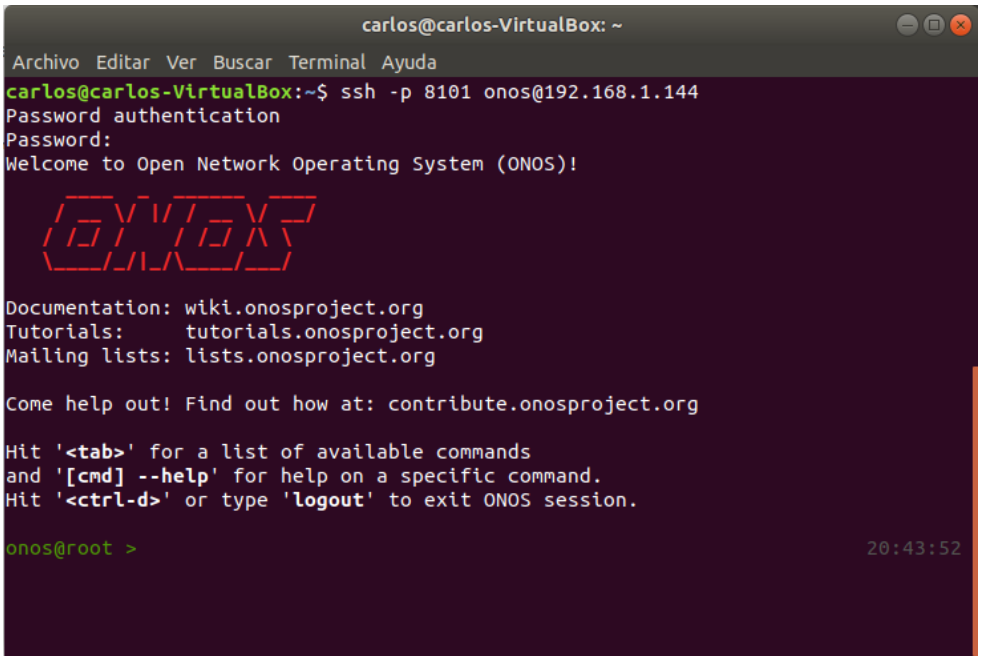


Figura 15: Interfaz CLI de ONOS

- **GUI (Entono gráfico):** Nos proporciona una visión de la red creada en ONOS con los switches y hosts que la conforman. Nos permitirá activar/desactivar aplicaciones y monitorizar el tráfico a tiempo real por los enlaces, en otras muchas funciones.

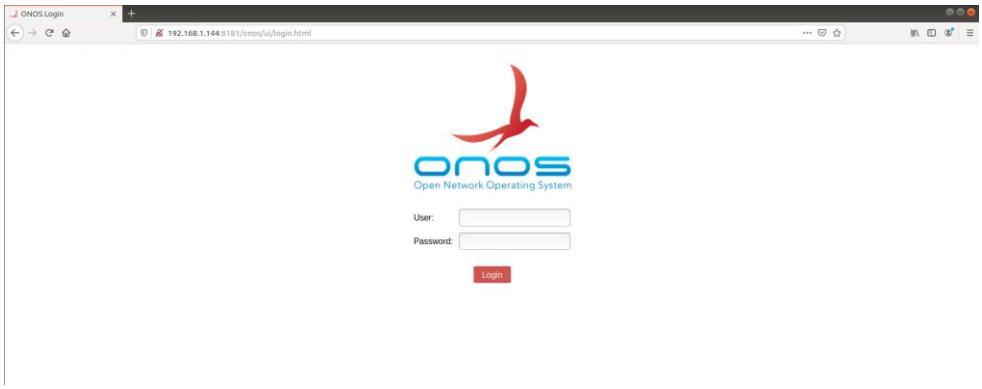


Figura 16: Interfaz GUI de ONOS

- **API REST:** una interfaz RESTful para la interfaz administrativa utilizada por la CLI

3.1.6 Versiones de ONOS

En la siguiente imagen se muestran las últimas versiones de ONOS desarrollas y que están disponibles para su descarga en https://wiki.onosproject.org/display/ONOS/Downloads.

Name	Version	JAVA API	Date
Velociraptor (LTS)	2.5.0	API-2.5.0	Dec 4,2020
Uguisu	2.4.0	API-2.4.0	Jun 5,2020
Toucan	2.3.0	API-2.3.0	Jan 27,2020
Sparrow (LTS)	2.2.0	API-2.2.0	Aug 30,2019
	2.2.1	API-2.2.1	Feb 20, 2020
	2.2.2	API-2.2.2	Mar 25, 2020
	2.2.3	API-2.2.3	Jul 3, 2020
Raven	2.1.0	API-2.1.0	Apr 30, 2019
Quail	2.0.0	API-2.0.0	Jan 18, 2019
Peacock (LTS)	1.15.0	API-1.15.0	Nov 29, 2018
Owl	1.14.0	API-1.14.0	Sep 4, 2018
Nightingale	1.13.10	API-1.13.10	Feb 20, 2020
	1.13.3	API-1.13.3	Sep 5, 2018
	1.13.2	API-1.13.2	July 11, 2018
	1.13.1	API-1.13.1	May 2, 2018
Magpie (LTS)	1.12.0	API-1.12.0	Dec 11, 2017
Loon	1.11.2	API-1.11.2	March 28, 2018
	1.11.1	API-1.11.1	Sep 15, 2017
	1.11.0	API-1.11.0	Sep 8, 2017
Kingfisher	1.10.4	API-1.10.4	Aug 25, 2017
	1.10.3	API-1.10.3	Aug 2, 2017
	1.10.2	API-1.10.2	Jun 22, 2017
	1.10.0	API-1.10.0	Jun 5, 2017

Figura 17: Versiones de ONOS disponibles

3.2 Mininet

Mininet [2][20][21] es una herramienta programada en Python cuyo objetivo es el de emular redes de telecomunicación. Permite crear redes con hosts, switches, controladores y enlaces a un alto nivel. Los hosts de Mininet corren bajo un sistema operativo Linux, mientras que los switches soportan el protocolo OpenFlow para mayor flexibilidad respecto a la configuración del routing y para integrarlos dentro de un escenario SDN.

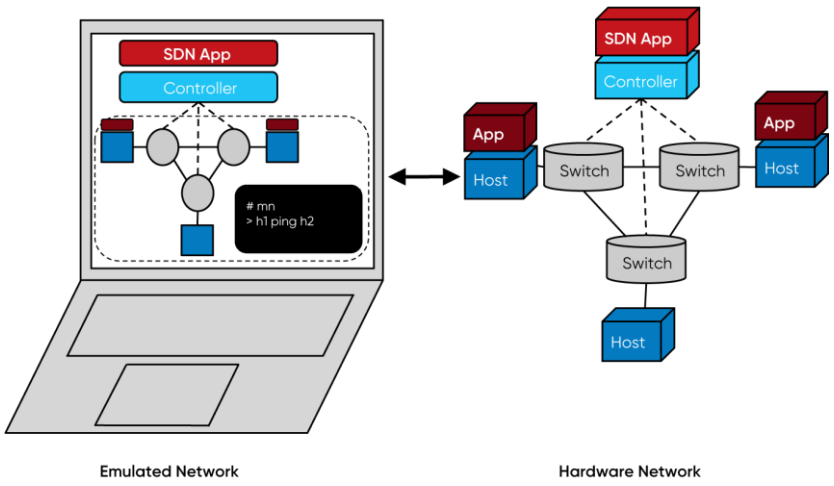


Figura 18: Emulación Mininet [21]

Sus principales características son:

- Es rápido. Implementar una red y probar su conectividad lleva apenas unos segundos.
- Se pueden ejecutar aplicaciones en cada host, individualmente. Desde un servidor web a herramientas de monitorización, como wireshark.
- Se pueden modelar las características de los enlaces: delay, baud-rate. Incluso se pueden modelar fallos de red conectando/desconectando enlaces.
- Permite que varios desarrolladores trabajen de forma concurrente sobre la misma topología de red
- Provee un amplio banco de pruebas para desarrollar aplicaciones basadas en Open-Flow.
- Permite realizar tests exhaustivos de topologías sin necesidad de tener una real.
- Permite crear desde topologías más sencillas con un único comando hasta topologías realmente complejas haciendo uso de una API programada en Python para definir los componentes con total detalle.

- Incluye una Interfaz de Línea de Comandos que es independiente de la topología emulada y del protocolo que ésta utilice.

Con solo un comando, Mininet puede crear una red virtual real en una máquina (VM, nube o local) en unos pocos segundos, ejecutando el kernel real, el conmutador y el código de la aplicación:

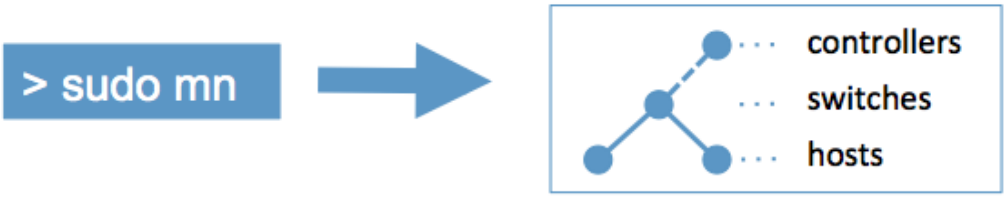


Figura 19: Comando básico para creación de red en Mininet [20]

La red simulada por Mininet ejecuta aplicaciones Linux estandarizadas, lo que permite que todo el desarrollo realizado y probado en Mininet se transfiera al sistema real con una mínima modificación.

3.2.1 Mininet – Wifi

Mininet-WiFi [10][11] es un emulador para redes inalámbricas código base tiene su origen en Mininet, y por lo tanto está desarrollado para el sistema Linux. Soporta WiFi nativo y además es capaz de simular otras tecnologías de red inalámbricas.

Mininet-WiFi
Emulator for Software-Defined Wireless Networks

A diferencia de Mininet, Mininet-WiFi aporta la posibilidad de virtualizar estaciones y puntos de acceso wifi además de los nodos de Mininet existentes, como hosts, conmutadores y controladores OpenFlow, y, no obstante, también permite el procesamiento de paquetes utilizando el protocolo OpenFlow.

3.2.2 MiniEdit

Se trata de una alternativa para la creación de topologías de red desde un entorno gráfico.

MiniEdit [11] está escrito en Python y fue desarrollado inicialmente para Mininet. No obstante, también ha sido actualizado para poder trabajar con Mininet-WiFi, donde el objetivo de los desarrolladores de MiniEdit fue habilitar todas las funcionalidades compatibles con Mininet-WiFi en MiniEdit.

Como se puede ver en la siguiente figura, MiniEdit tiene una interfaz de usuario bastante simple que presenta una pantalla con una línea de iconos de herramientas en el lado izquierdo de la ventana y una barra de menú en la parte superior.

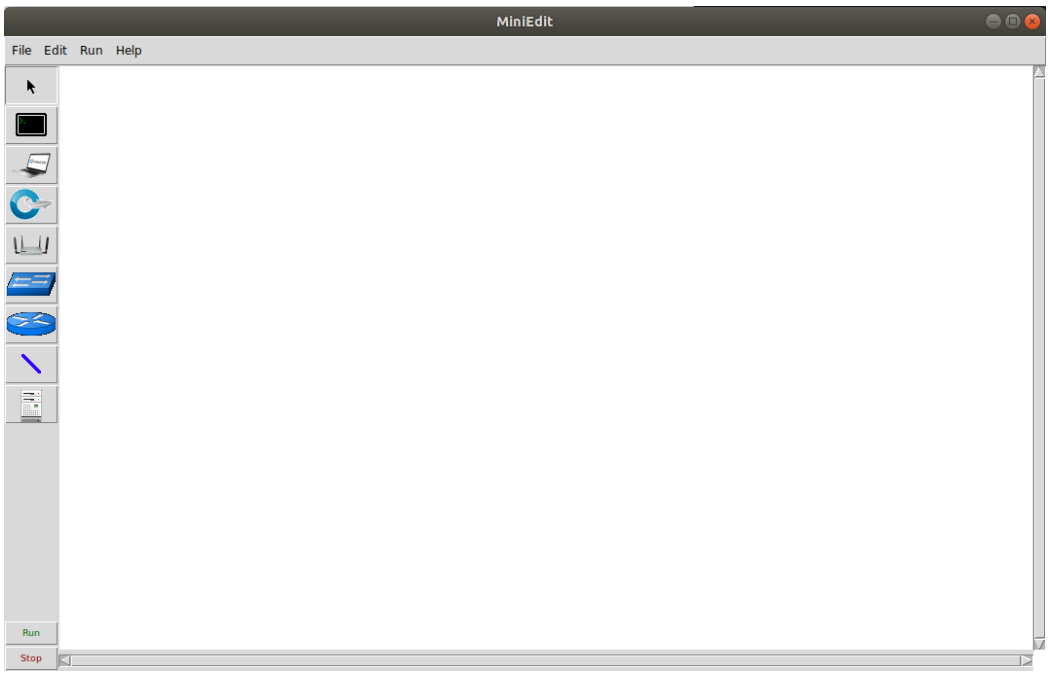


Figura 20: MiniEdit

Para lanzar la herramienta de miniedit debemos introducir el siguiente comando desde la carpeta examples de mininet-wifi:

- /mininet-wifi/examples$ sudo ./miniedit

Como se puede observar, esta herramienta da la posibilidad de añadir distintos elementos al diseño de red, modificar configuraciones de estos y agregar los enlaces correspondientes. Permite incluir elementos como:

- Host
- Estaciones de trabajo wifi

- Switches openFlow
- Puntos de acceso Wifi
- Switches sin openFlow
- Router
- Controlador SDN

Cabe destacar, aunque no se hará uso de estos en este TFM, que en la versión actual de MiniEdit ya se admiten diferentes tipos de escenarios como, por ejemplo: ad hoc y redes mesh, WiFi-Direct, protocolo Radius, WPA, etc.

3.2.3 Comandos básicos de Mininet

A continuación, se detallan los principales comandos que se emplean en Mininet y el uso que se le dan.

- **dpctl**
  es una utilidad de administración que permite cierto control sobre el switch OpenFlow, Permite agregar flujos a las tablas, consultar sus características y cambiar otras configuraciones.

- **dump**
  muestra la información detallada de la red, como los dispositivos, nombre, puertos, direcciones ip,etc.

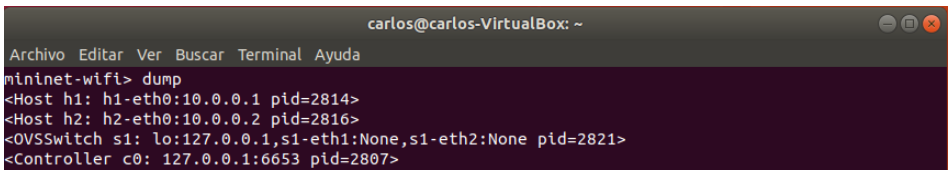


Figura 21: Mininet - Salida comando 'dump'

- **xterm**
  este comando abre una ventana de línea de comandos de un nodo específico.

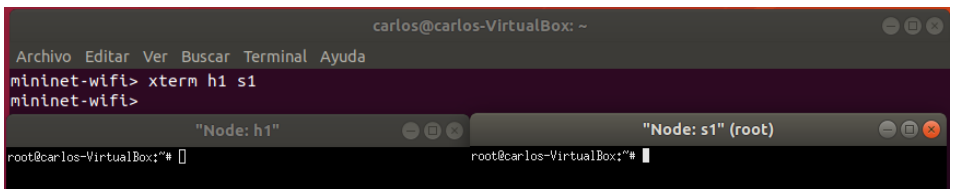


Figura 22: Mininet - Salida comando 'xterm'

- **help**
  muestra ayuda e información sobre uso de comandos.

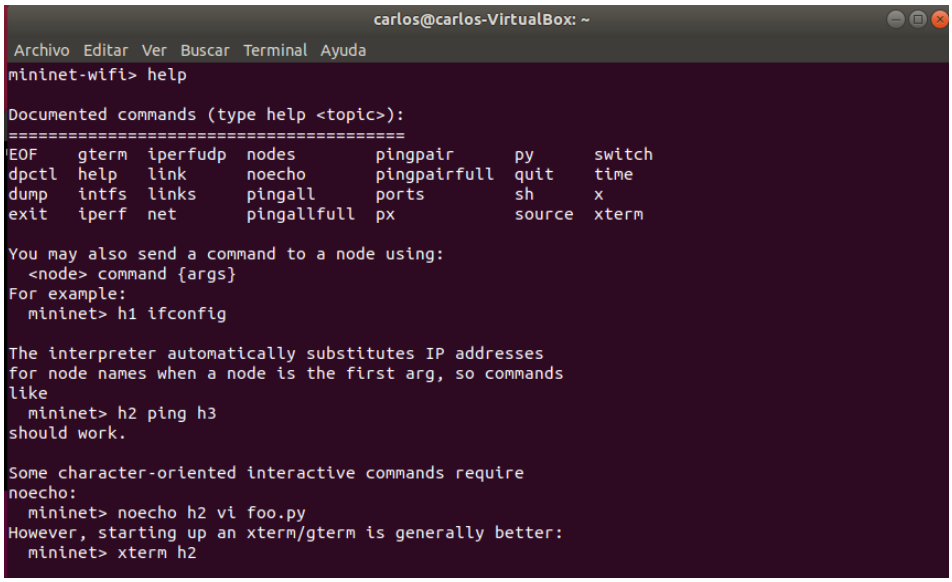


Figura 23: Mininet - Salida comando 'help'

- **intfs**
  muestra una lista de las interfaces de los diferentes nodos.

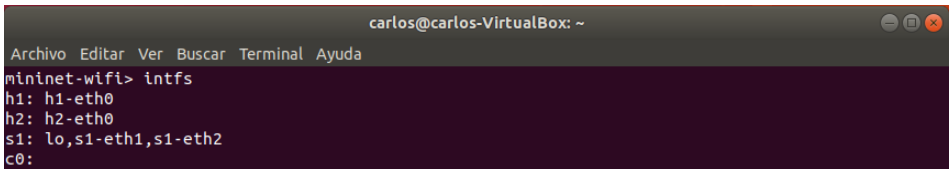


Figura 24: Mininet - Salida comando 'intfs'

- **iperf**
  realiza una prueba simple de TCP entre dos hosts, donde se indica el ancho de banda máximo entre los hosts indicados.

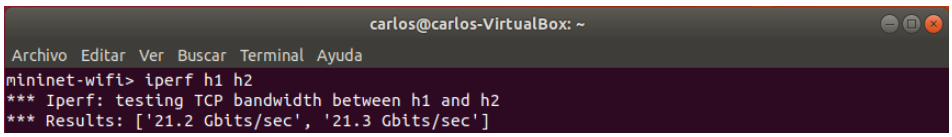


Figura 25: Mininet - Salida comando 'ipref'

- **iperfudp**
  realiza una prueba semejante a la anterior, pero por UDP.

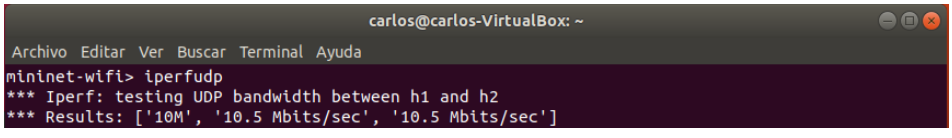


Figura 26: Mininet - Salida comando 'iperfudp'

- **link**
  para habilitar/deshabilitar el enlace entre dos nodos

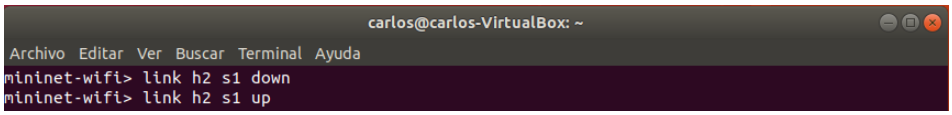


Figura 27: Mininet - Salida comando 'link'

- **links**
  muestra el estado de todos los enlaces operativos.

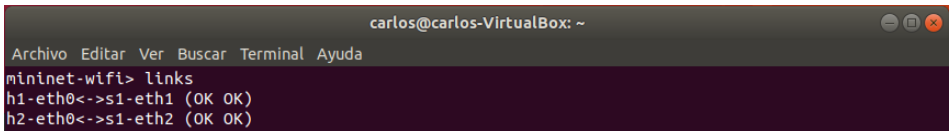


Figura 28: Mininet - Salida comando 'links'

- **net**
  muestra los enlaces y los puertos utilizados.

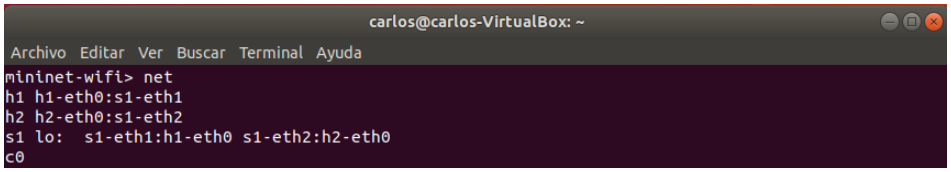


Figura 29: Mininet - Salida comando 'net'

- **nodes**
  muestra una lista de todos los nodos de la red.

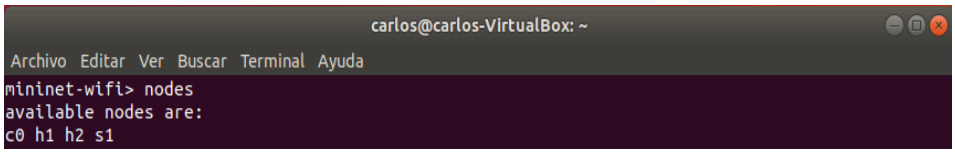


Figura 30: Mininet - Salida comando 'nodes'

- **pingall**
  realiza una prueba de ping entre todos los hosts, devolviendo un resumen de todos los pings realizados.

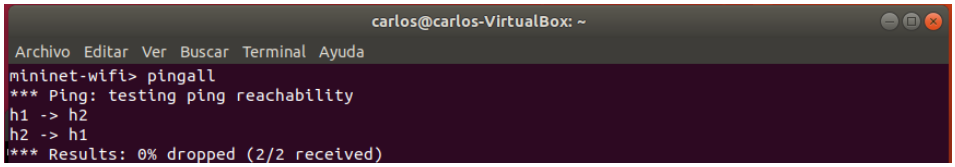


Figura 31: Mininet - Salida comando 'pingall'

- **pingallfull**
  realiza un ping ente todos los hosts, devolviendo todos los resultados detallados de las operaciones.

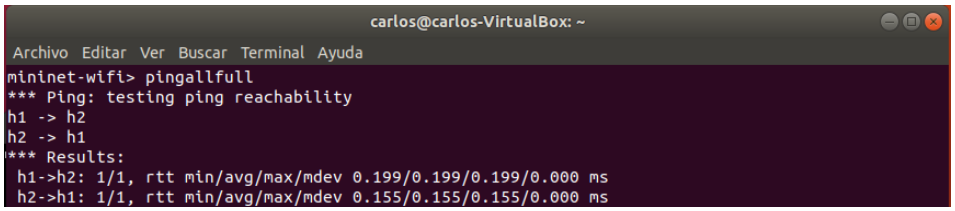


Figura 32: Mininet - Salida comando 'pingallfull'

- **ports**
  muestra la información de los puertos y las interfaces de los diferentes switches.

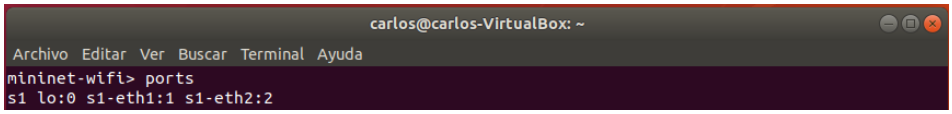


Figura 33: Mininet - Salida comando 'ports'

- **switch**
  se emplea para arrancar o parar un switch.

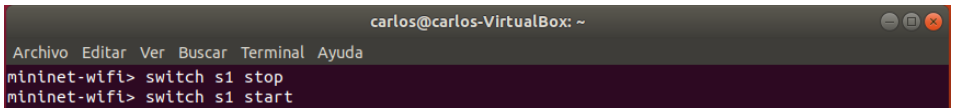


Figura 34: Mininet - Salida comando 'ports'

4. Estudio modelo de red actual de un hospital

Para este apartado nos apoyaremos en el Trabajo Final de Máster del Ignacio Campos Marcé con título 'Diseño de una organización Sanitaria" [22].

4.1 Estudio de la topología

Para el estudio de esta nos basaremos en el diseño de modelo de red que planteó Ignacio Campos.

En este modelo observamos una arquitectura dividida en 3 capas o niveles de servicio (CORE, DISTRIBUCIÓN Y ACCESO) y 4 bloques (CORE, WAN, USUARIOS y CPD).

En la siguiente figura se puede observar el modelo de red planteado con las distintas capas y bloques mencionados anteriormente.

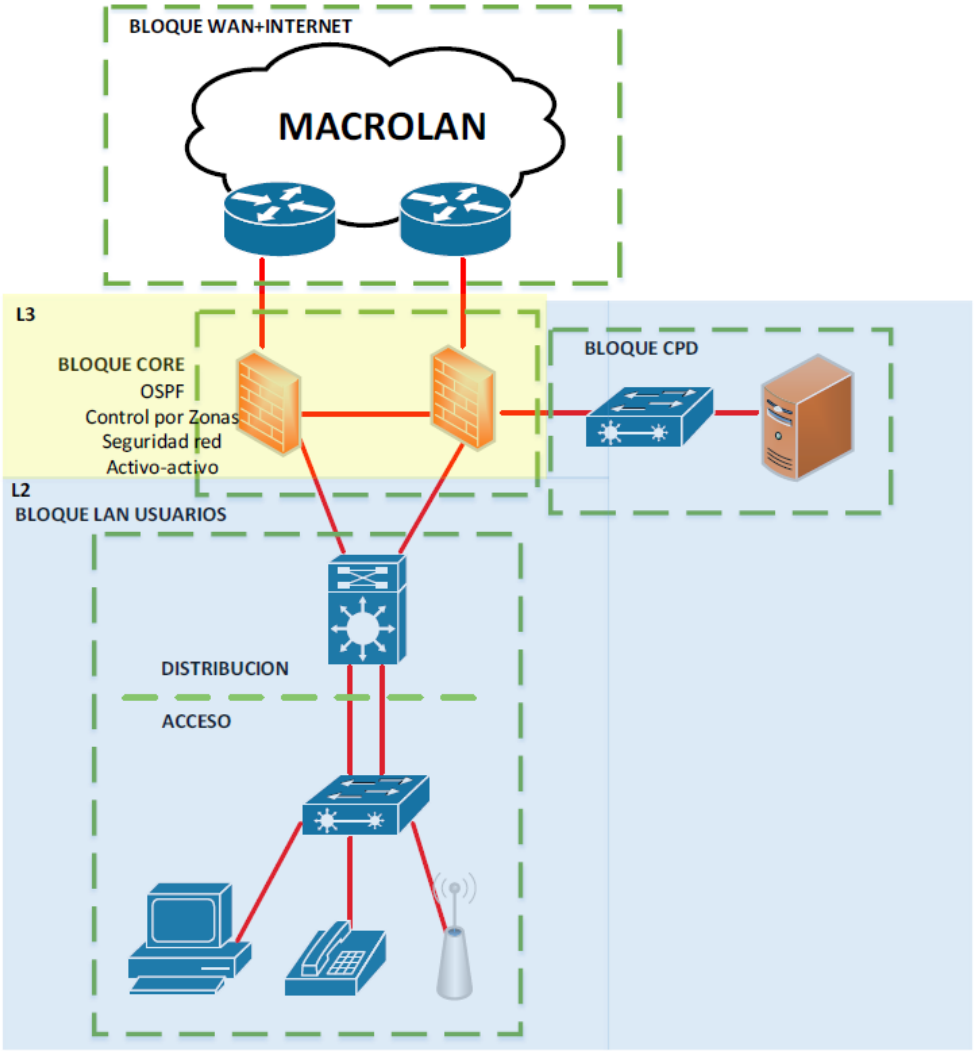


Figura 35: Modelo de red Hospitales [22]

4.1.1 Bloque Core

Este bloque está compuesto por dos firewalls que actúan de nivel 3 y configurados en modo activo-activo. Todo el tráfico circula a través de ellos convirtiéndose así en una arquitectura de Core compacto con OSPF.

Estos dispositivos dotan a los administradores de red de la capacidad de controlar todo el tráfico del hospital y permite realizar separaciones de la red por zonas, aumentando de esta manera la seguridad.

4.1.2 Bloque Internet

Este bloque está compuesto por los routers que dan acceso a la red de la comunidad autónoma y a internet. Al mismo tiempo la central telefónica también se encuentra conectada en este bloque.

Estos routers conectan con los firewalls perimetrales (Nivel 3 del hospital), donde se aplican reglas de capa 7 y se activan las funciones de IPS e inspección de antivirus dotando así de una barrera de protección a la sede frente a posibles amenazas.

4.1.3 Bloque CPD

Este bloque está segmentado por 3 zonas o entornos destinados a TEST, PRE y PRO, que hacen referencia a las distintas fases de puesta en marcha de nuevas aplicaciones.

Este bloque alberga servidores que alojan servicios propios del hospital para el uso de los profesionales sanitarios de la misma sede, por ello se descartó la posibilidad de implantar un balanceador de carga.

Los switches de este bloque conectan directamente con los firewalls de CORE que son los encargados de gestionar los accesos a los recursos del CPD.

4.1.4 Bloque Usuarios

Este bloque está formado por dos capas, capa de falso distribución y capa de acceso.

La capa de falso distribución recibe este nombre debido a que anteriormente albergaba el nivel 3 del hospital y en la nueva topología se

traslada a los firewalls de Core. Esta capa está formada por dos switches Cisco 4500 en stack y conecta con cada armario del hospital con dos fibras de 10Gb/s y se forma un port-channel dotando así de un enlace con capacidad de 20Gb/s por armario.

De la misma que en distribución, en los armarios de acceso se pueden encontrar hasta 4 switches Cisco 2960 o 9200 en stack. Ante la necesidad de instalar más switches en el armario para dotar de red a más usuarios, según normativa implantada se debe tirar dos nuevas fibras hasta el mismo para poder conectar 4 switches más en stack. A continuación, se muestra las pautas a seguir:

Switches por Armario	Números de fibras
1-4	2
5-8	4
9-12	6
12-16	8

Tabla 1: Relación de Switches y fibras por armario

En la siguiente imagen podemos ver el diseño lógico planteado en un hospital.

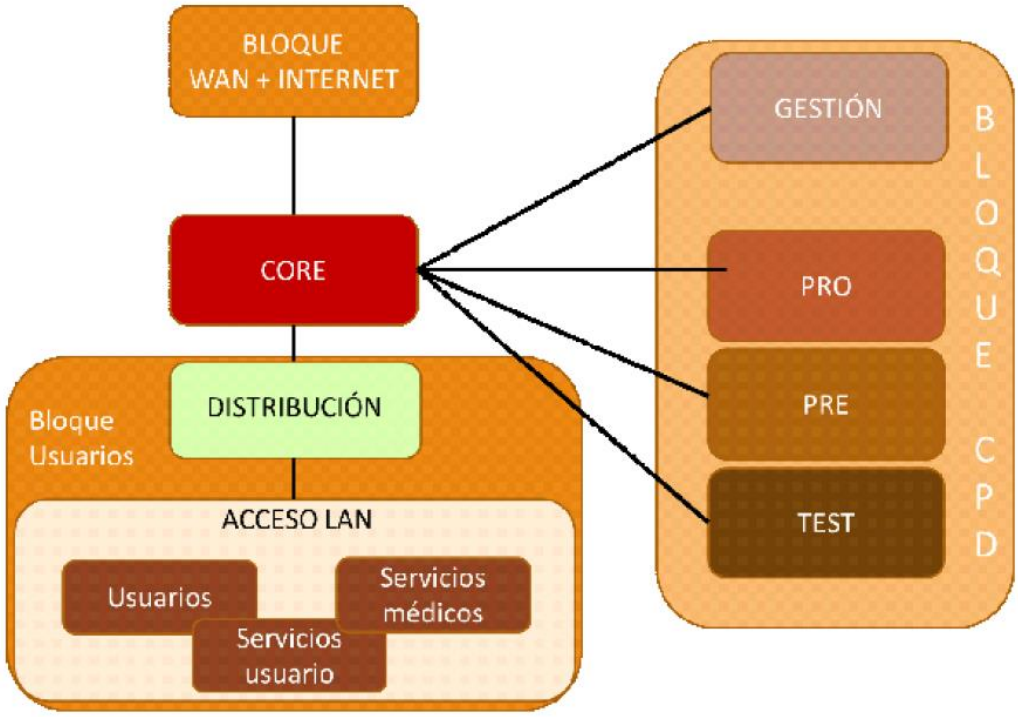


Figura 36: Diseño lógico Hospital [22]

Cabe destacar que la electrónica que compone la infraestructura de red es compatible con OpenFlow, por lo que no sería necesaria una

inversión para poder adaptar la infraestructura a la hora de implantar SDN.

4.2 Segmentación por zonas y VLAN`s desplegadas

Se realiza una segmentación de la red por zonas en el firewall de core disponiendo un conjunto de grupos diferenciados y se realiza una separación de dominios broadcast mediante la creación de VLAN's definidas según el rol del dispositivo o usuario.

4.2.1 Zona Usuarios

Esta zona albergará a usuarios y dispositivos de administración donde se definen las siguientes VLAN's:

- **VLAN 10**: Para usuarios con equipos dentro del dominio de la organización
- **VLAN 11**: Para usuarios con equipos fuera del dominio de la organización
- **VLAN 12:** Para impresoras.
- **VLAN 13:** Multifunción con escáner o sólo escáner.
- **VLAN 15:** Para equipos de urgencias.
- **VLAN 16:** Equipos de personal de informática con acceso al resto de redes locales del hospital.
- **VLAN 17:** Equipos clientes ligeros
- **VLAN 90:** Para telefonía IP o equipos de videoconferencia.
- **VLAN 95:** Plan B alternativo, grupo de equipos de cada una de las VLAN's definidas para usuarios que deberían de tener disponibilidad máxima.
- **VLAN 98:** Para los puertos libres de los switches que no estén en uso.
- **VLAN 99:** Para todos los puertos en modo trunk que no necesiten una VLAN nativa.

4.2.2 Zona Gestión de red (Gest_red)

Esta zona se destina a la administración del equipamiento de red donde se definen las siguientes VLAN's:

- **VLAN 20**: Para la gestión de las controladoras y los puntos de acceso wifi instalados en el centro.
- **VLAN 21**: para la gestión electrónica de red (Switches)

4.2.3 Zona Servicios de Medicina (Serv_Medicina)

Esta zona se destina a equipamientos qué ofrecen servicios médicos donde se definen las siguientes VLAN's:

- **VLAN 30**: Para equipos de radiología
- **VLAN 31**: Para analizadores de laboratorio.
- **VLAN 32:** Para dispositivos de electromedicina
- **VLAN 33:** Para dispositivos de monitorización de pacientes
- **VLAN 34:** Para localización de pacientes o equipos.
- **VLAN 35:** Para escáner de muestras al microscopio.

4.2.3 Zona Servicios externos (Serv_Externos)

Esta zona se destina al equipamiento de servicios externos que no forman parte de los servicios médicos donde se definen las siguientes VLAN's:

- **VLAN 40**: Para equipos de mantenimiento en el centro como: Aires acondicionados, alumbrado, SAI's, sensores de temperatura, sensores de humos, alarmas, megafonía.
- **VLAN 41**: Para la gestión de cámaras de seguridad.
- **VLAN 42:** Para la gestión de dispositivos de control de acceso a determinadas zonas del hospital.
- **VLAN 43:** Para dispositivos que identifiquen stock, inventario o logística de la sede.
- **VLAN 44:** Para dispositivos sin clasificar como, por ejemplo, kioscos monitor de citas

4.2.4 Zona Servicios Wifi (Serv_Wifi)

Esta zona se destina a equipamiento de servicios wifi donde se definen las siguientes VLAN's:

- **VLAN 110**: Para usuarios con equipos dentro del dominio de la organización que requieran de conexión vía wifi.
- **VLAN 116**: Para con equipos fuera del dominio de la organización que requieran de conexión vía wifi
- **VLAN 130:** Para equipos de radiología que se conectan vía wifi.

- **VLAN 133:** Para dispositivos de monitorización de pacientes que se conectan vía wifi.
- **VLAN 166:** Para el servicio Wifi a ciudadanos. Está separada del resto de VLAN's, está limitada a 5 usuarios concurrentes por punto acceso wifi y emplea una salida a internet independiente al hospital.
- **VLAN 190:** Para equipos de telefonía IP inalámbrica.

4.2.5 Zona CPD

Esta zona se destina a equipamiento alojado en el CPD del hospital donde se definen las siguientes VLAN's:

- **VLAN 210**: Para servidores en el entono de TEST
- **VLAN 220**: Para servidores en el entono de PRE
- **VLAN 230:** Para servidores en el entono de PRO

4.3 Electrónica de red actual en los Hospitales

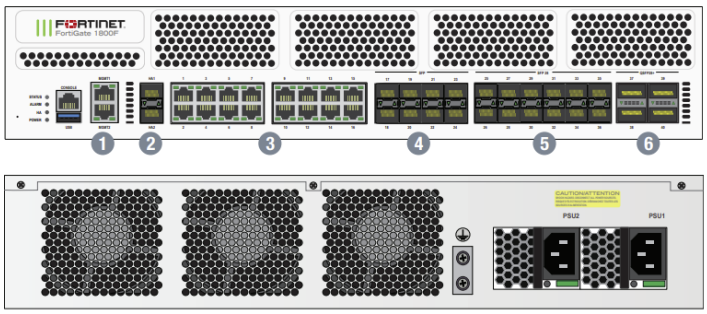
4.3.1 Bloque Core

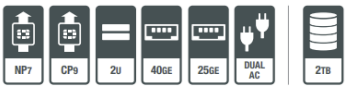
Como ya se ha mencionado, este bloque está compuesto por dos firewalls que actúan de nivel 3 y configurados en modo activo-activo. Todo el tráfico circula a través de ellos convirtiéndose así en una arquitectura de Core compacto con OSPF.

Actualmente este bloque consta de dos Firewalls Fortinet FG-1800F Series. En las siguientes imágenes se muestran sus especificaciones.

No obstante, este bloque no se tendrá en cuenta para el diseño de red planteando.

FortiGate 1800F Series





Interfaces

1. 2 x GE RJ45 MGMT Ports
2. 2 x 10 GE SFP+ / GE SFP HA Slots
3. 16 x GE RJ45 Ports
4. 8 x GE SFP Slots
5. 12 x 25 SFP28 / 10 GE SFP+ / GE SFP Slots
6. 4 x 40 GE QSFP+ Slots

	FG-1800F	FG-1801F
Hardware Specifications		
Hardware Accelerated GE RJ45 Ports	16	
Hardware Accelerated GE SFP Slots	8	
Hardware Accelerated 25 GE SFP28 / 10 GE SFP+ / GE SFP Slots	12	
Hardware Accelerated 40GE QSFP+ Slots	4	
GE RJ45 Management Ports	2	
10 GE SFP+ / GE SFP HA Slots	2	
USB 3.0 Port	1	
Console RJ45 Port	1	
Onboard Storage	-	2x 1TB NVMe SSD
Included Transceivers	2x SFP+ (SR 10GE)	
System Performance — Enterprise Traffic Mix		
IPS Throughput [2]	13 Gbps	
NGFW Throughput [2, 4]	11 Gbps	
Threat Protection Throughput [2, 5]	9.1 Gbps	
System Performance and Capacity		
IPv4 Firewall Throughput (1518 / 512 / 64 byte, UDP)	198 / 197 / 140 Gbps	
IPv6 Firewall Throughput (1518 / 512 / 86 byte, UDP)	198 / 197 / 140 Gbps	
Firewall Latency (64 byte, UDP)	3.22 µs	
Firewall Throughput (Packet per Second)	210 Mpps	
Concurrent Sessions (TCP)	12 Million	
New Sessions/Second (TCP)	750,000	
Firewall Policies	100,000	
IPsec VPN Throughput (512 byte) [1]	55 Gbps	
Gateway-to-Gateway IPsec VPN Tunnels	20,000	
Client-to-Gateway IPsec VPN Tunnels	100,000	
SSL-VPN Throughput	11 Gbps	
Concurrent SSL-VPN Users (Recommended Maximum, Tunnel Mode)	10,000	
SSL Inspection Throughput (IPS, avg. HTTPS) [3]	17 Gbps	
SSL Inspection CPS (IPS, avg. HTTPS) [3]	9,500	
SSL Inspection Concurrent Session (IPS, avg. HTTPS) [3]	1.3 Million	
Application Control Throughput (HTTP 64K) [2]	34 Gbps	
CAPWAP Throughput (HTTP 64K)	26.5 Gbps	
Virtual Domains (Default / Maximum)	10 / 250	
Maximum Number of FortiSwitches Supported	196	
Maximum Number of FortiAPs (Total / Tunnel)	4,096 / 2,048	
Maximum Number of FortiTokens	20,000	
High Availability Configurations	Active-Active, Active-Passive, Clustering	

	FG-1800F	FG-1801F
Dimensions and Power		
Height x Width x Length (inches)	3.5 x 17.25 x 21.1	
Height x Width x Length (mm)	88.4 x 438 x 536	
Weight	30.2 lbs (13.7 kg)	30.4 lbs (13.8 kg)
Form Factor (supports EIA/non-EIA standards)	Rack Mount, 2RU	
Power Source	100–240VAC, 50-60 Hz	
Maximum Current	7A@100VAC, 3A@240VAC	
Maximum Power Consumption	543.6W	558.8W
Average Power Consumption	388.3W	399.1W
Heat Dissipation	1,854.84 BTU/hr	1,906.70 BTU/hr
Redundant Power Supplies	Yes, Hot swappable	
Operating Environment and Certifications		
Operating Temperature	32–104°F (0–40°C)	
Storage Temperature	-31–158°F (-35–70°C)	
Humidity	10–90% non-condensing	
Noise Level	62.74 dBA	
Operating Altitude	Up to 7,400 ft (2,250 m)	
Compliance	FCC Part 15 Class A, RCM, VCCI, CE, UL/cUL, CB	
Certifications	ICSA Labs: Firewall, IPsec, IPS, Antivirus, SSL-VPN; USGv6/IPv6	

Figura 37: Bloque Core – Especificaciones Fortinet FG-1800F Series [13]

4.3.2 Bloque Internet

Como ya se comentó anteriormente, este bloque está compuesto por los routers que dan acceso a la red de la comunidad autónoma y a internet. Al mismo tiempo la central telefónica también se encuentra conectada en este bloque.

Estos servicios actualmente están gestionados por un proveedor de servicios externos y desde el Centro de Gestión de la Red Arterias no poseen permisos para administrar dichos elementos.

No obstante, este bloque no se tendrá en cuenta para el diseño de red planteando.

4.3.3 Bloque CPD

Como ya se comentó anteriormente, este bloque está segmentado por 3 zonas o entornos destinados a TEST, PRE y PRO, que hacen referencia a las distintas fases de puesta en marcha de nuevas aplicaciones.

Los switches de este bloque conectan directamente con los firewalls de CORE que son los encargados de gestionar los accesos a los recursos del CPD.

Los elementos de red que conforman este bloque son Cisco Catalyst 2960-X Series Switches [14] de 24 y 48 puertos. Estos switches son compatibles con OpenFlow por lo que no será necesaria una inversión en equipamiento nuevo para su adaptación a las SDN.

Figura 38: Bloque CPD - Cisco Catalyst 2960-X Series Switches [14]

En la web del fabricante Cisco [15] se pueden observar las funciones de OpenFlow 1.3 y OpenFlow 1.0 que son compatibles, los Match/Actions/Pipelines admitidos y los comandos de configuración de los switches para configurar OpenFlow.

Feature	Notes
Configuration of physical interfaces as OpenFlow logical switch ports	Bridge domain, Virtual LANs and Virtual Routing and Forwarding (VRF), and port-channel interfaces are not supported. Only L2 interfaces can be OpenFlow logical switch ports.
Supported OpenFlow message types	Controller to switch: • Handshake • Switch Configuration • Modify State (Port Modification message is not supported) • Read State • Packet-Out • Barrier Asynchronous messages: • Packet-In • Flow Removed • Port Status • Error Symmetric messages: • Hello • Echo Request • Echo Reply • Vendor
Connection to controllers	You can connect up to eight controllers. Connection to the controller through a management interface or a switched virtual interface (SVI) is supported. Connection via TCP and TLS is supported.
Multiple actions	If multiple actions are associated with a flow, they are processed in the order specified. The output action should be the last action in the action list. Any action after the output action is not supported, and can cause the flow to fail and return an error to the controller. Flows defined on the controller must follow the these guidelines: • The flow can have only one output action. • Some action combinations which are not supported may be rejected at flow programming time. • The flow should not have an output-to-controller action in combination with other rewrite actions.
Supported OpenFlow counters	Per Table–Active entries, packet lookups, and packet matches. Per Flow–Received Packets, Received bytes, Duration (seconds), Duration (milliseconds). Per Port–Received or transmitted packets, and bytes. Per Controller– Flow addition, modification, deletion, error messages, echo requests or replies, barrier requests or replies, connection attempts, successful connections, packet in or packet out.
Default forwarding rule	All packets that cannot be matched to programmed flows are dropped by default. You can configure sending unmatched packets to the controller. You can modify the default action taken on unmatched packets either using the default-miss command or by the controller.
Idle timeout	A minimum Idle timeout of 14 seconds is supported for 700 flows and 48 ports. The statistics collection interval influences the minimum idle timeout. When the interval is set to 7 seconds, the timeout is a minimum of 14 seconds. 700 flows are supported with the 14-second idle timeout. When using an idle timeout of less than 25 seconds, the number of L3 flows should be limited to 700.

Tabla 2: Compatibilidad switch Cisco Catalyst 2960-X con funciones OpenFlow [15]

Feature	Notes
Pipelines	Pipelines are mandatory for logical switch. The logical switch supports only pipeline 1. The logical switch supports only table 0.
Forwarding Table	Match Criteria: • Input Port • Ethernet type • Source Mac Address • Dest Mac Address • VLAN ID • IP TOS (DSCP bits) • IP Protocol (except for lower 8 bits of ARP code) • IPv4 Source Address • IPv4 Destination Address • Layer 4 Source Port • Layer 4 Destination Port • IPv6 Source Address • IPv6 Destination Address Action Criteria: • Forward: Controller • Forward: Port • Forward: Drop • Forward: Controller + Port • Set VLAN ID • New VLAN ID • Replace VLAN ID • Strip VLAN Header • Modify Source MAC • Modify Destination MAC • Modify IPv4 Source Address • Modify IPv4 Destination Address • Modify IPv4 TOS bits • Modify L4 source port • Modify L4 destination port • Decrement TTL
Number of flows	1000
Configuration of VLANs	VLAN range is from 1 to 4094.

Tabla 3: Match/Actions/Pipelines admitidos por switch Cisco Catalyst 2960-X [15]

4.3.4 Bloque Usuarios

Como ya se comentó anteriormente, este bloque está formado por dos capas, capa de falso distribución y capa de acceso.

La capa de falso distribución recibe este nombre debido a que anteriormente albergaba el nivel 3 del hospital, y está formada por switches Cisco Catalyst 4500-X Series [16], en concreto el modelo Cisco Catalyst 4500X-32 SFP+ Switch con puertos SFP+ a 10 Gigabit Ethernet. Serán compatibles con OpenFlow instalando Cisco Plug-in for OpenFlow [17] como se indica en la web del fabricante, no siendo necesario invertir en equipamiento nuevo para su adaptación a las SDN.

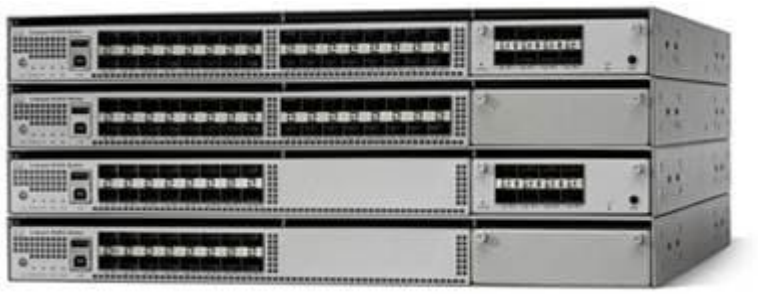


Figura 39: Bloque Usuarios – Falso distribución - Cisco Catalyst 4500-X Series [16]

En cuanto a los armarios de acceso, estos están compuestos por Cisco Catalyst 2960-X Series Switches [14] de 24 y 48 puertos PoE+, ya comentados en el Bloque de CPD, y Cisco Catalyst 9200 Series Switches [18] de 24 y 48 puertos PoE+.

Figura 40: Bloque Usuarios – Acceso - Cisco Catalyst 9200-X Series [18]

Revisando el datasheet de los Cisco Catalyst 9200 Series Switches [18] y la guía de configuración de IOS XE [19] se comprueba que actualmente los Cisco Catalyst 9200 Series no son compatibles con OpenFlow.

Por lo tanto, sería necesaria la inversión en electrónica de red nueva para poder adaptar en su totalidad los armarios de acceso de los hospitales al diseño planteado para las SDN con el controlador ONOS.

5. Diseño de red propuesto

Teniendo presente el principal objetivo de este TFM que era el análisis y estudio de las redes definidas por software (SDN) para su implantación en el entorno sanitario y dotar a las redes de los hospitales de una mayor eficiencia, flexibilidad y escalabilidad. Una vez estudiadas las herramientas de las que harán uso y realizar el estudio del modelo de red de un hospital que plantea Ignacio Campos en su TFM [22], es hora de detallar el diseño a implementar para cumplir los objetivos del trabajo.

Cabe destacar que la mayor parte de la actual electrónica de red que conforma la topología de los hospitales es compatible con OpenFlow a excepción del Cisco Catalyst 9200 Series, por lo tanto, se reduce considerablemente la inversión necesaria para la implantación de las SDN con las herramientas planteadas.

Dado el alto número de VLAN's planteadas en el modelo de red para hospitales, se decide realizar la simulación y pruebas del diseño con un número más reducido. Se emplearán la VLAN 10 (Usuarios), VLAN 30 (Radiología), VLAN 90(VOZ), VLAN 110(Usuarios Wifi), VLAN 210 (TEST_CPD).

Para ello se hará uso de la controladora ONOS, instalando la última versión Velociraptor (LTS) 2.5.0 y las herramientas Mininet y Mininet-Wifi para emular el diseño de red.

5.1 Instalación de ONOS

Como ya se ha mencionado anteriormente, se hará uso del controlador SDN ONOS (Open Network Operating System) para realizar las pruebas del diseño propuesto.

Para ello se empleará el entorno Virtual Box donde se instalará una máquina virtual con el sistema operativa Ubuntu 18.04 con los siguientes requisitos mínimos de hardware:

- 2 core CPU
- 2 GB RAM
- 10 GB hdd
- 1 NIC (any speed)

No se pretende entrar en detalle de los comandos a introducir para la instalación de ONOS, ya que se puede encontrar una guía de instalación en https://wiki.onosproject.org/.

Una vez instalado el controlador ONOS y todos los complementos necesarios para su correcto funcionamiento, a continuación, se explica la

manera de activar/desactivar aplicaciones de ONOS según las funcionalidades que se quieran implementar en el controlador.

Como ya se ha comentado en el anterior capítulo 3 sobre ONOS, es posible gestionar el controlador de tres maneras distintas: CLI (Línea de comandos), GUI (Entorno gráfico) y API REST.

A continuación, se explicará la manera de revisar los servicios activos y activar o desactivarlos mediante CLI y GUI.

- **Mediante CLI**

  Al inroducir el comando que se muestra en la siguiente figura se mostrarán los servicios activos actualmente en el controlador.

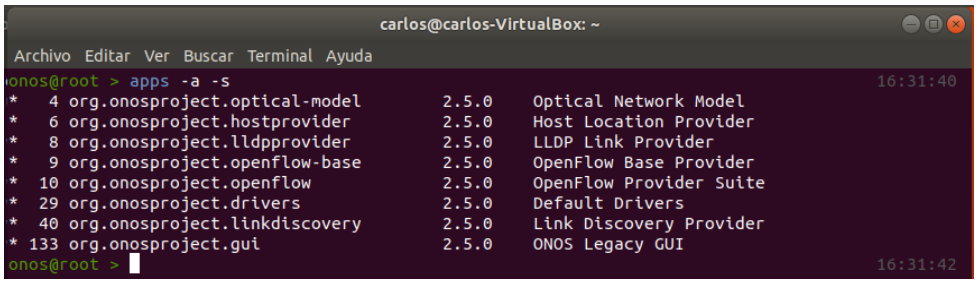


Figura 41: ONOS - Consulta servicios activos CLI

Si se introduce el comando que se muestra en la siguiente figura se podrá activar/desactivar un servicio.

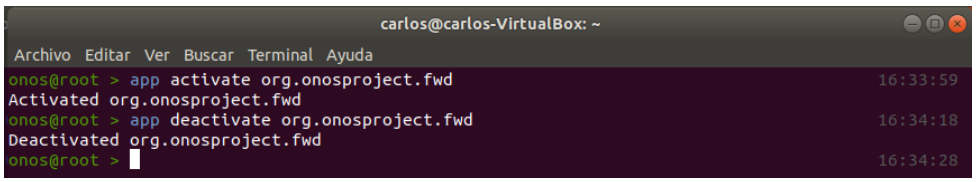


Figura 42: ONOS – Activación/Desactivación de servicios CLI

- **Mediante GUI**

  En el apartado de Aplicaciones se pueden consultar los servicios activos ✔ y activar o desactivarlos como se muestra en la siguiente figura.

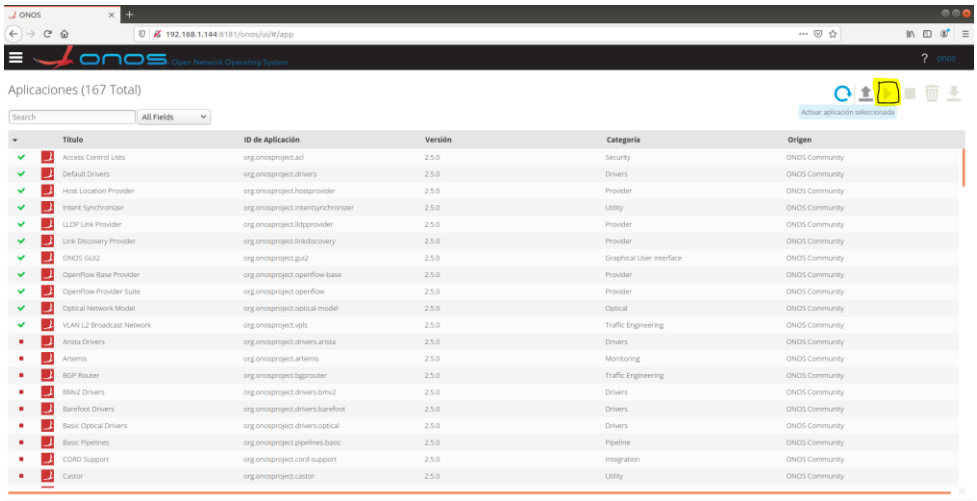


Figura 43: Consulta y activación/desactivación de servicios onos (GUI)

5.2 Instalación de Mininet-Wifi

Sobre la misma máquina virtual con SO Ubuntu 18.04 se debe instalar openvswitch y mininet-wifi para poder llevar a cabo las simulaciones previstas.

No se pretende entrar en detalle de los comandos a introducir para la instalación de Mininet-Wifi, ya que se puede encontrar una guía de instalación en https://mininet-wifi.github.io/get-started/.

Se ha optado por emplear Mininet-Wifi para realizar el diseño de red ya que esta herramienta permite añadir elementos inalámbricos al diseño, como, por ejemplo: puntos de acceso wifi y estaciones de trabajo o equipos portátiles.

Como ya se ha comentado anteriormente en el Capítulo 3, la creación de topologías de red mediante Mininet o Mininet-Wifi puede realizarse empleando un entorno gráfico llamado MiniEdit. Dada la simplicidad que aporta este entorno, se ha escogido MiniEdit para realizar el diseño porque además de permitir realizarlo de una manera gráfica, permite exportarlo a Python para posteriormente lanzar la simulación.

5.3 Diseño de red

En este apartado de presentará el diseño propuesto tras el estudio de la topología de red realizado en el capítulo 4.

No obstante, como se puede observar en la siguiente imagen, respecto a la topología de red estudiada se mantiene intactos los bloques WAN y CORE. Este primero, debido que en la actualidad este bloque es gestionado por una empresa externa a través de una licitación pública, y

el segundo porque los Firewalls perimetrales implantados dotan a los hospitales de una alta protección contra amenazas.

Por consiguiente, se plantea la implantación del controlador ONOS de manera centralizada para actuar sobre los bloques de CPD y USUARIOS de todos hospitales.

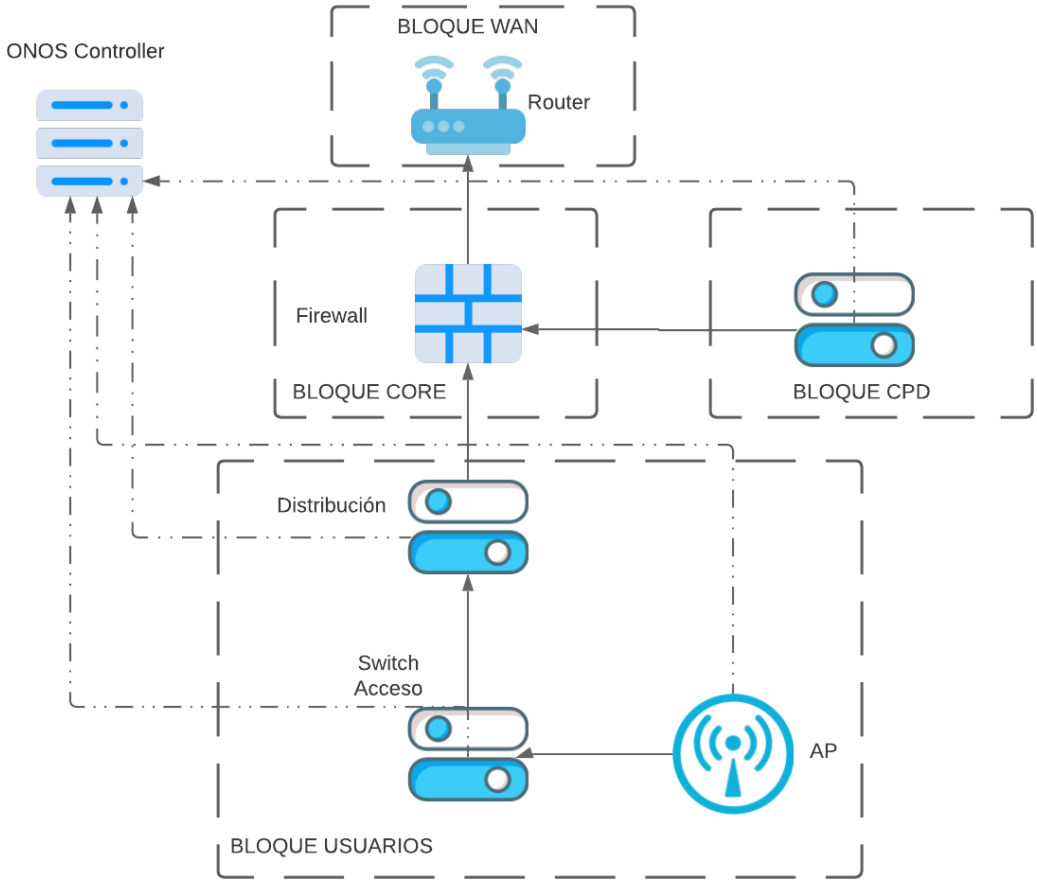


Figura 44: Diseño topología de red hospitales para SDN

5.4 Diseño para monitorización de red

En este apartado se tratará el diseño planteado para llevar a cabo las simulaciones para la monitorización de tráfico. Como ya se comentado anteriormente, hoy en día no se dispone de ninguna herramienta de monitorización que de una manera rápida y visual identifique un elemento que causa saturación en la red de la sede.

Tomando como base la anterior topología de red expuesta en la Figura 43 y teniendo en cuenta la reducción de VLAN's mencionada al comienzo del capítulo 5, se plantea la topología que se muestra en la siguiente figura.

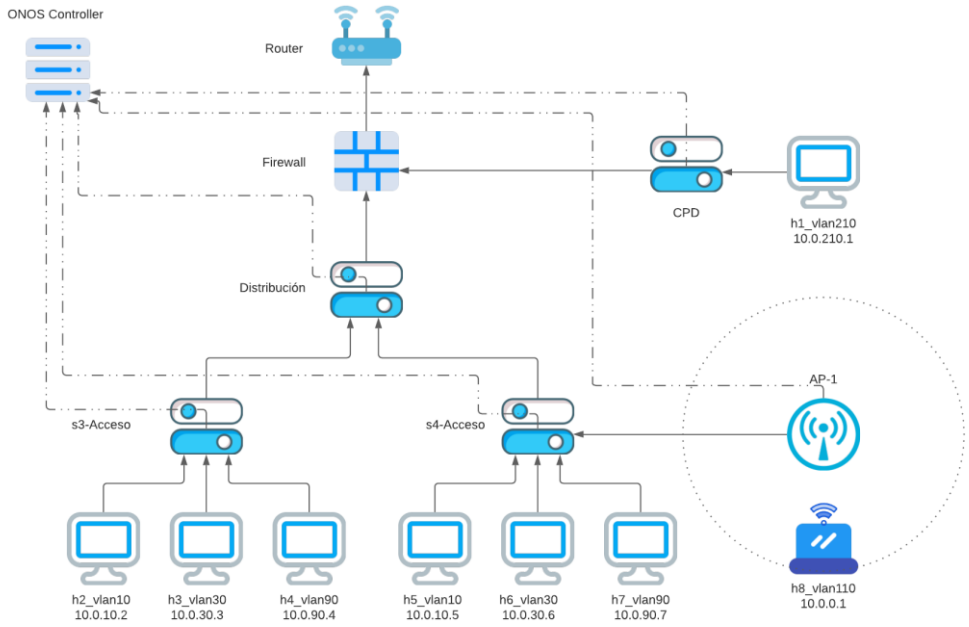


Figura 45: Topología de red para monitorización de Tráfico en Hospital

Como se puede observar, el diseño consta de los siguientes elementos que entrarán en acción en la simulación:

- Un controlador ONOS
- 4 Switches compatibles con OpenFlow
- Un punto de acceso Wifi
- 7 hosts y una estación de trabajo Wifi

En la siguiente tabla se adjunta la información referente a la configuración de los hosts:

Host	Dirección ip	VLAN	VLAN name
h1	10.0.210.1	210	TEST_CPD
h2	10.0.10.2	10	Usuarios
h3	10.0.30.3	30	Radiología
h4	10.0.90.4	90	Voz
h5	10.0.10.5	10	Usuarios
h6	10.0.30.6	30	Radiología
h7	10.0.90.7	90	Voz
h8(sta1)	10.0.0.1	110	Usuarios Wifi

Tabla 4: Configuración hosts diseño de red

A continuación, se realiza la adaptación de la topología propuesta para su implementación en MiniEdit y posterior simulación.

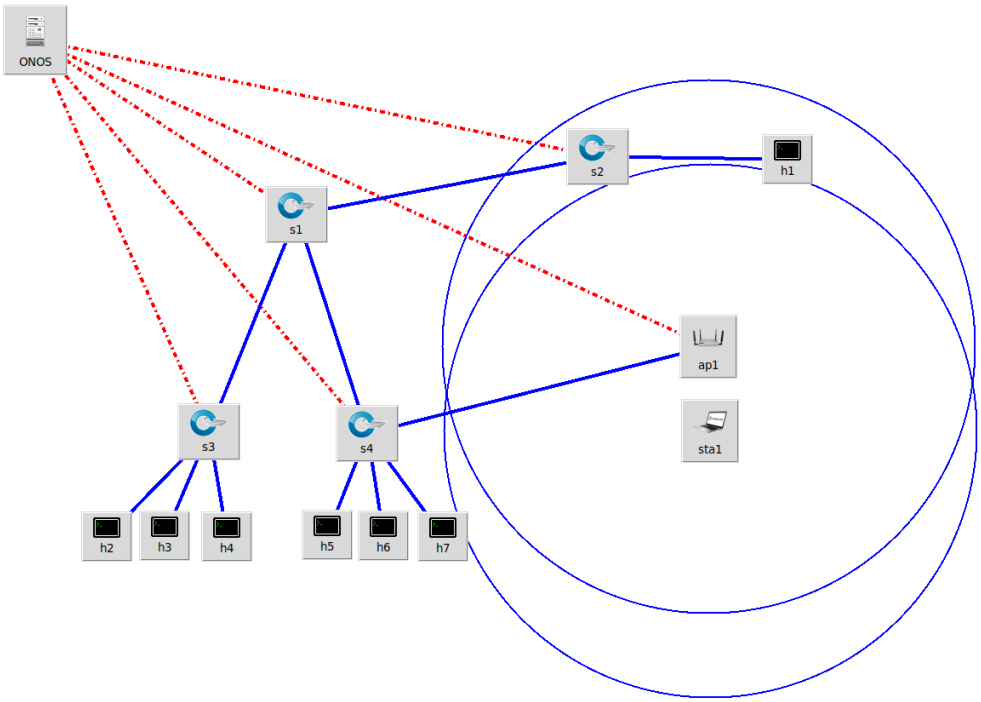


Figura 46: Miniedit: Diseño de red para monitorización de Tráfico en Hospital

El código resultante del diseño se puede consultar en el anexo A.

Como se aprecia en la siguiente figura, una de las muchas ventajas que ofrece MiniEdit es que permite configurar los parámetros del controlador remoto ONOS y las ip's asignadas al resto de elementos que conforman el diseño.

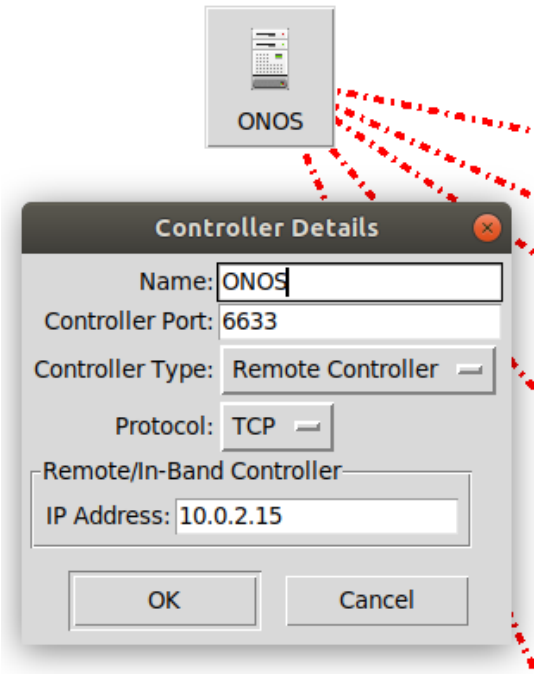


Figura 47: Miniedit: Configuración controlador remoto

Una vez finalizada la configuración de los parámetros necesarios se carga el diseño, y como se muestra en la siguiente figura, la carga del diseño es satisfactoria.

```
mininet-wifi> nodes
available nodes are:
ONOS ap1 h1 h2 h3 h4 h5 h6 h7 s1 s2 s3 s4 sta1
mininet-wifi>
mininet-wifi>
mininet-wifi>
mininet-wifi> links
h2-eth0<->s3-eth1 (OK OK)
h3-eth0<->s3-eth2 (OK OK)
h4-eth0<->s3-eth3 (OK OK)
h5-eth0<->s4-eth1 (OK OK)
h6-eth0<->s4-eth2 (OK OK)
h7-eth0<->s4-eth3 (OK OK)
ap1-eth2<->s4-eth4 (OK OK)
s3-eth4<->s1-eth1 (OK OK)
s4-eth5<->s1-eth2 (OK OK)
s1-eth3<->s2-eth1 (OK OK)
s2-eth2<->h1-eth0 (OK OK)
sta1-wlan0<->wifi (OK wifi)
```

Figura 48: Mininet-wifi: Carga de diseño satisfactoria - monitorización de Tráfico en Hospital

Se debe revisar si el diseño se ha cargado satisfactoriamente en ONOS.

Se verifica en las siguientes figuras, donde se aprecia la carga en el controlador ONOS con los hosts y devices registrados.

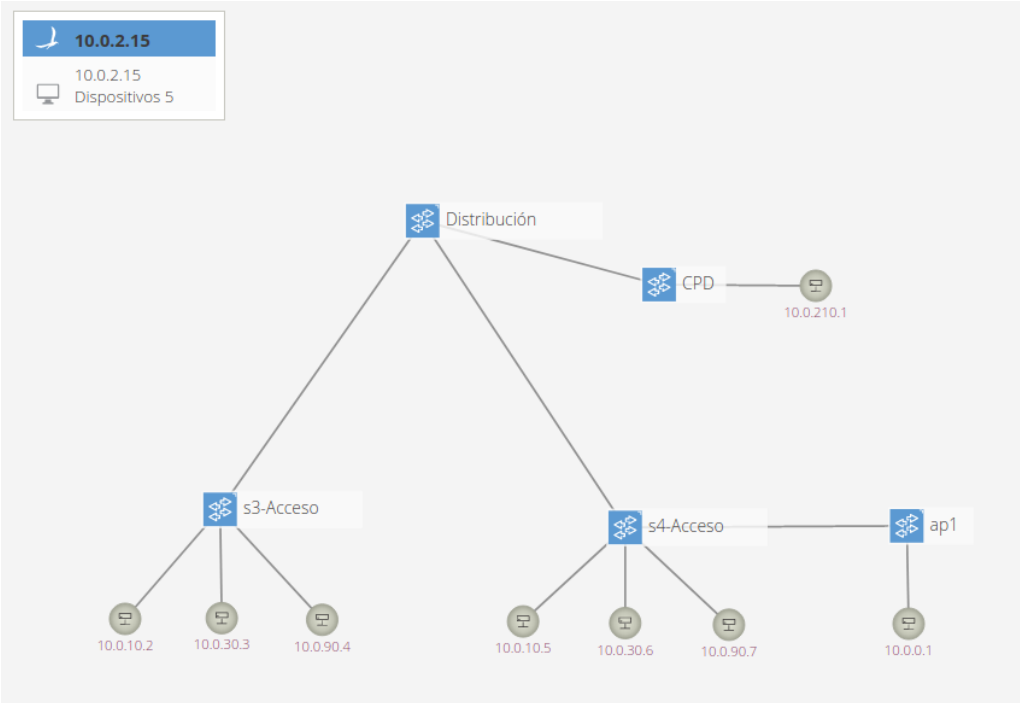


Figura 49: ONOS GUI: Carga de diseño satisfactoria - monitorización de Tráfico en Hospital

```
onos@root > hosts                                                    19:40:00
id=02:00:00:00:00:00/None, mac=02:00:00:00:00:00, locations=[of:1000000000000001/1], auxLocations=null, vlan=Non
e, ip(s)=[10.0.0.1], innerVlan=None, outerTPID=unknown, provider=of:org.onosproject.provider.host, configured=fa
lse
id=3A:1F:21:0E:C7:E0/None, mac=3A:1F:21:0E:C7:E0, locations=[of:0000000000000004/1], auxLocations=null, vlan=Non
e, ip(s)=[10.0.10.5], innerVlan=None, outerTPID=unknown, provider=of:org.onosproject.provider.host, configured=f
alse
id=3A:CA:1F:F0:84:BB/None, mac=3A:CA:1F:F0:84:BB, locations=[of:0000000000000004/3], auxLocations=null, vlan=Non
e, ip(s)=[10.0.90.7], innerVlan=None, outerTPID=unknown, provider=of:org.onosproject.provider.host, configured=f
alse
id=6A:7F:13:11:C6:3D/None, mac=6A:7F:13:11:C6:3D, locations=[of:0000000000000002/2], auxLocations=null, vlan=Non
e, ip(s)=[10.0.210.1], innerVlan=None, outerTPID=unknown, provider=of:org.onosproject.provider.host, configured=
false
id=6E:DC:6F:36:20:56/None, mac=6E:DC:6F:36:20:56, locations=[of:0000000000000003/3], auxLocations=null, vlan=Non
e, ip(s)=[10.0.90.4], innerVlan=None, outerTPID=unknown, provider=of:org.onosproject.provider.host, configured=f
alse
id=9A:E4:B2:C4:37:4E/None, mac=9A:E4:B2:C4:37:4E, locations=[of:0000000000000004/2], auxLocations=null, vlan=Non
e, ip(s)=[10.0.30.6], innerVlan=None, outerTPID=unknown, provider=of:org.onosproject.provider.host, configured=f
alse
id=9E:F4:66:F0:A4:EF/None, mac=9E:F4:66:F0:A4:EF, locations=[of:0000000000000003/2], auxLocations=null, vlan=Non
e, ip(s)=[10.0.30.3], innerVlan=None, outerTPID=unknown, provider=of:org.onosproject.provider.host, configured=f
alse
id=AE:2B:E9:D3:2A:85/None, mac=AE:2B:E9:D3:2A:85, locations=[of:0000000000000003/1], auxLocations=null, vlan=Non
e, ip(s)=[10.0.10.2], innerVlan=None, outerTPID=unknown, provider=of:org.onosproject.provider.host, configured=f
alse
onos@root >                                                          19:40:03
onos@root >                                                          19:40:25
onos@root > devices                                                  19:40:25
id=of:0000000000000001, available=true, local-status=connected 6m29s ago, role=MASTER, type=SWITCH, mfr=Nicira,
Inc., hw=Open vSwitch, sw=2.9.5, serial=None, chassis=1, driver=ovs, channelId=10.0.2.15:37372, locType=none, ma
nagementAddress=10.0.2.15, name=Distribución, protocol=OF_14
id=of:0000000000000002, available=true, local-status=connected 6m30s ago, role=MASTER, type=SWITCH, mfr=Nicira,
Inc., hw=Open vSwitch, sw=2.9.5, serial=None, chassis=2, driver=ovs, channelId=10.0.2.15:37374, locType=none, ma
nagementAddress=10.0.2.15, name=CPD, protocol=OF_14
id=of:0000000000000003, available=true, local-status=connected 6m30s ago, role=MASTER, type=SWITCH, mfr=Nicira,
Inc., hw=Open vSwitch, sw=2.9.5, serial=None, chassis=3, driver=ovs, channelId=10.0.2.15:37368, locType=none, ma
nagementAddress=10.0.2.15, name=s3-Acceso, protocol=OF_14
id=of:0000000000000004, available=true, local-status=connected 6m29s ago, role=MASTER, type=SWITCH, mfr=Nicira,
Inc., hw=Open vSwitch, sw=2.9.5, serial=None, chassis=4, driver=ovs, channelId=10.0.2.15:37376, locType=none, ma
nagementAddress=10.0.2.15, name=s4-Acceso, protocol=OF_14
id=of:1000000000000001, available=true, local-status=connected 6m30s ago, role=MASTER, type=SWITCH, mfr=Nicira,
Inc., hw=Open vSwitch, sw=2.9.5, serial=None, chassis=1000000000000001, driver=ovs, channelId=10.0.2.15:37370, l
ocType=none, managementAddress=10.0.2.15, name=ap1, protocol=OF_14
onos@root >                                                          19:40:30
```

Figura 50: ONOS CLI: Carga de diseño satisfactoria - monitorización de Tráfico en Hospital

Tras preparar el entorno para la simulación, procedemos a realizar la prueba de monitorización. Cabe destacar que para realizar esta prueba se ha habilitado la app de ONOS de Reactive Forwarding.

Se genera tráfico mediante el comando de Mininet 'iperf', donde se calculará el ancho de banda máximo entre los hosts h2 y h1. Como muestra la imagen, se verifica que los parámetros adquiridos en ambas herramientas son muy similares.

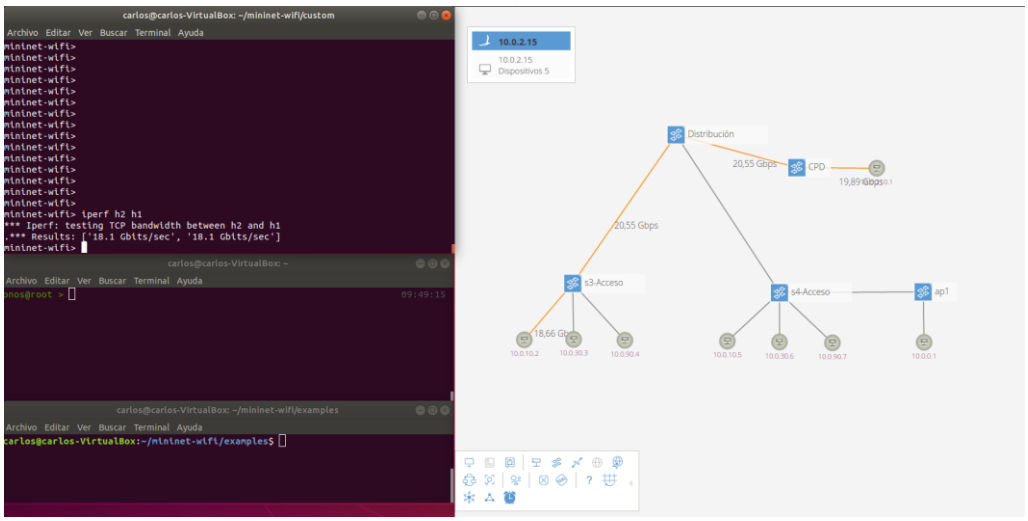


Figura 51: Resultado prueba Monitorización de tráfico Mininet - ONOS

Como se aprecia en la figura anterior, Mininet indica una media de ancho de banda 18.1 Gbits/s y ONOS muestran valores con un mínimo de 18.66 Gbits/s y un pico de 20.55 Gbits/s.

Por consiguiente, teniendo en cuenta los resultados obtenidos por ONOS se puede concluir que el margen de error obtenido en los resultados obtenidos es mínimo y que la funcionalidad de monitorización que incorpora ONOS es una gran alternativa para cubrir la carencia que actualmente sufren los administradores de red para detectar un equipo que satura la red en la sede.

5.5 Diseño para implantación de VPLS

En este apartado del capítulo 5 se pretende abordar el diseño de red para cumplir con el objetivo de implantación de Virtual Private LAN Service (VPLS) en ONOS.

Como ya se ha comentado anteriormente, se desea implementar VPLS para habilitar la comunicación entre equipos de la misma VLAN y que se encuentran ubicados en hospitales distintos De esta manera se conseguirá que todos los hospitales parezcan estar en la misma LAN Ethernet y se aportará mayor seguridad a la red.

Para llevar a cabo la configuración de VPLS se empleará la topología que se muestra en la siguiente imagen, donde se indican las distintas VPLS a crear en ONOS.

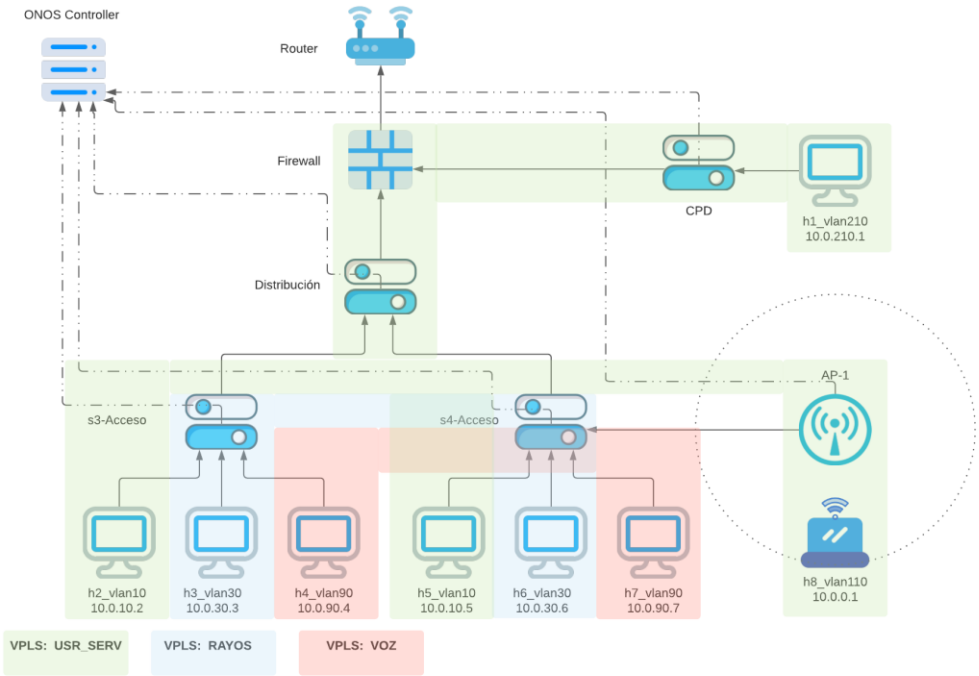


Figura 52: Diseño General de red para configuración de VPLS

En la siguiente tabla de expone de manera resumida la configuración de las distintas VPLS

VPLS	host	Dirección ip	VLAN
USR_SERV	h1	10.0.210.1	210
	h5	10.0.10.5	10
	h2	10.0.10.2	10
	h8(sta1)	10.0.0.1	110
RAYOS	h3	10.0.30.3	30
	h6	10.0.30.6	30
VOZ	h7	10.0.90.7	90
	h4	10.0.90.4	90

Tabla 5: Configuración VPLS planteada

Se crea VPLS USR_SERV, para unificar en un misma VPLS los usuarios tanto de conexión wifi como cableada y los servicios web o recursos que puedan tener alojados los hospitales en sus CPD's.

El código resultante del diseño es el mismo que el anterior apartado y se puede consultar en el anexo A.

Para llevar a cabo la siguiente simulación será necesario tener activas las aplicaciones del ONOS que se muestran en la siguiente figura.

```
onos@root > apps -a -s
*   4 org.onosproject.optical-model        2.5.0    Optical Network Model
*   6 org.onosproject.hostprovider         2.5.0    Host Location Provider
*   8 org.onosproject.lldpprovider         2.5.0    LLDP Link Provider
*   9 org.onosproject.openflow-base        2.5.0    OpenFlow Base Provider
*  10 org.onosproject.openflow             2.5.0    OpenFlow Provider Suite
*  16 org.onosproject.intentsynchronizer   2.5.0    Intent Synchronizer
*  29 org.onosproject.drivers              2.5.0    Default Drivers
*  40 org.onosproject.linkdiscovery        2.5.0    Link Discovery Provider
*  84 org.onosproject.gui2                 2.5.0    ONOS GUI2
*  87 org.onosproject.vpls                 2.5.0    VLAN L2 Broadcast Network
onos@root >
```

Figura 53: Apps necesarias de ONOS para configurar VPLS

Una vez lanzado el diseño y se ha comprobado que se ha cargado satisfactoriamente, se comienza con la configuración de VPLS.

Como se puede observar en la siguiente figura, si se lanza una prueba de conectividad antes de comenzar con la implementación de VPLS, no existe comunicación entre ninguno de los hosts.

```
mininet-wifi> pingallfull
*** Ping: testing ping reachability
h5 -> X X X X X X X
h2 -> X X X X X X X
h1 -> X X X X X X X
h6 -> X X X X X X X
h7 -> X X X X X X X
h3 -> X X X X X X X
h4 -> X X X X X X X
sta1 -> X X X X X X X
```

Figura 54: Prueba de conectividad fallida VPLS

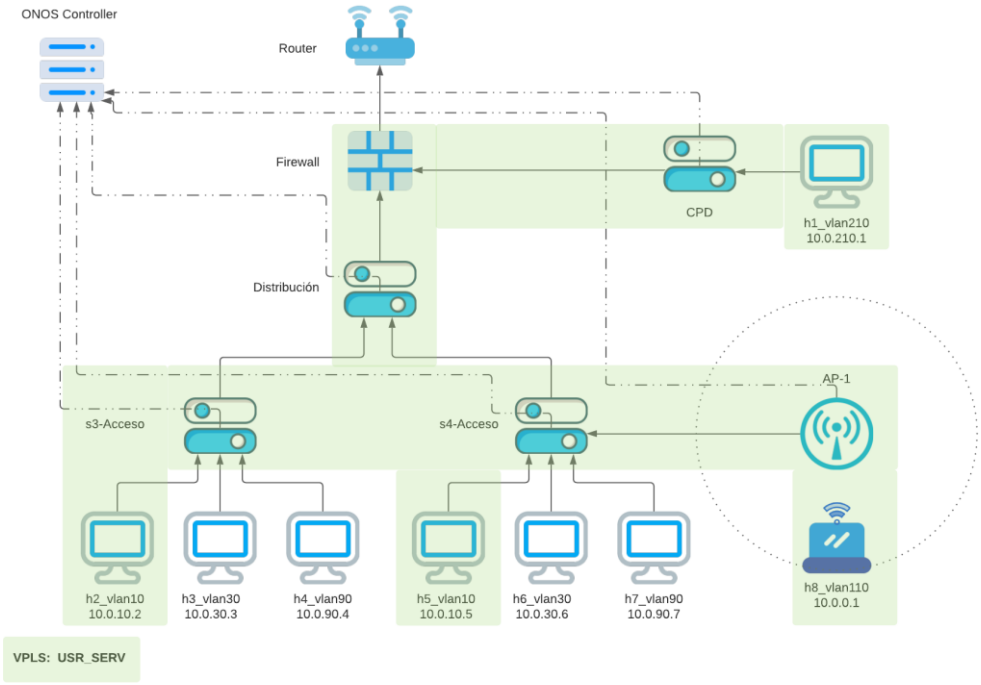


Figura 55: Diseño VPLS USR_SERV

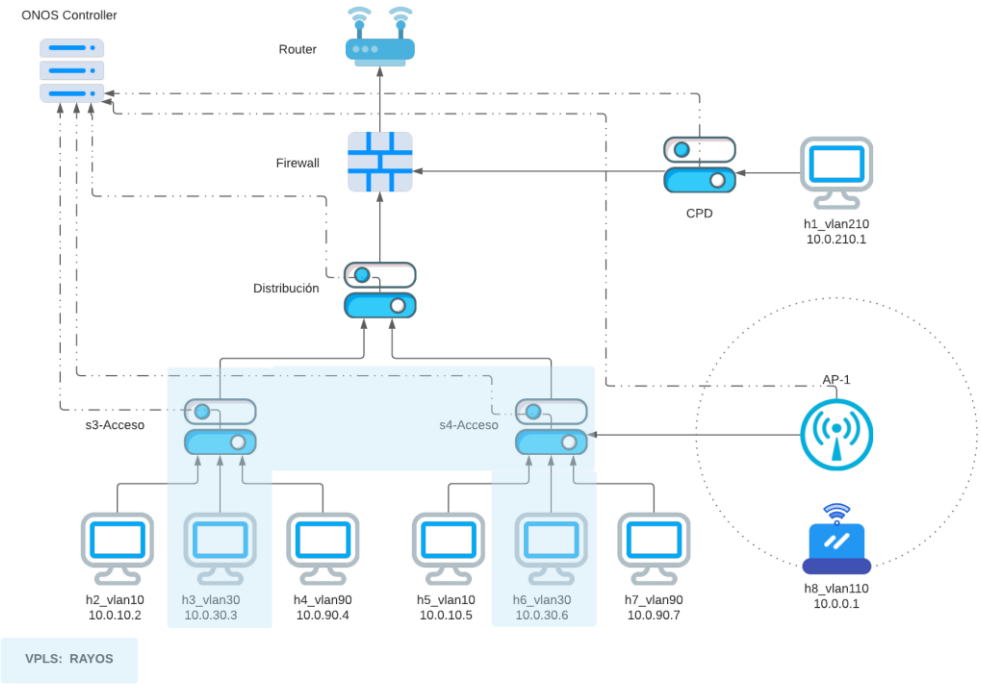


Figura 56: Diseño VPLS RAYOS

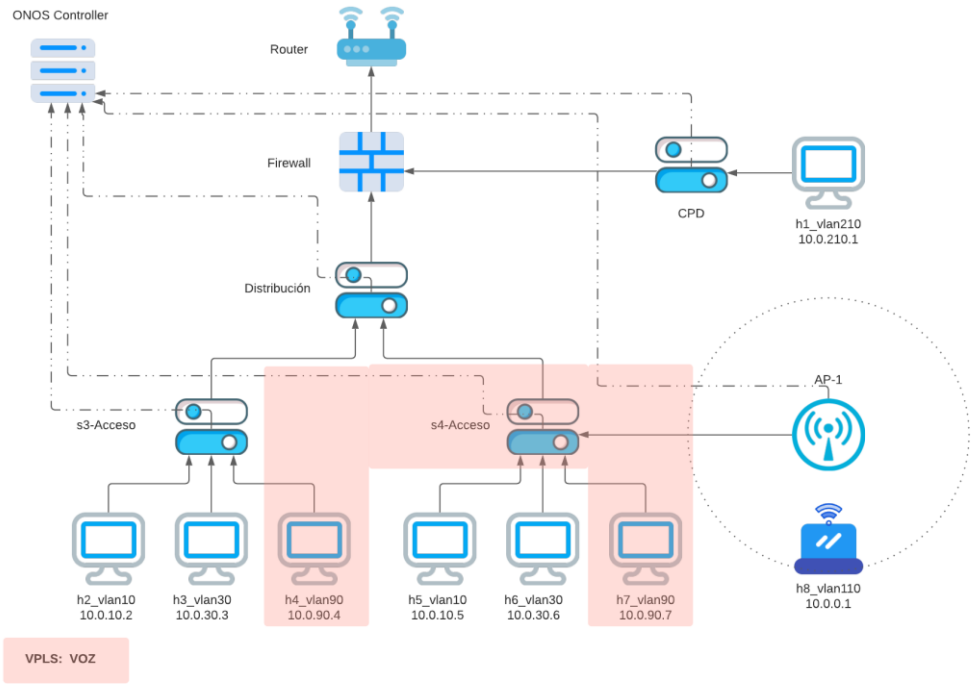


Figura 57: Diseño VPLS VOZ

Para la configuración de la VPLS se seguirán los siguientes pasos de implementación en ONOS:

- Se crearán las interfaces asociadas a los hosts
- Se crearán las VPLS indicadas en la tabla 5
- Se asignarán los hosts a las VPLS indicadas según la tabla 5

```
onos@root > interface-add of:0000000000000002/2 h1
> interface-add of:0000000000000003/1 h2
> interface-add of:0000000000000003/2 h3
> interface-add of:0000000000000003/3 h4
> interface-add of:0000000000000004/1 h5
> interface-add of:0000000000000004/2 h6
> interface-add of:0000000000000004/3 h7
> interface-add of:1000000000000001/1 sta1
Interface added
Interface added
Interface added
Interface added
Interface added
Interface added
Interface added
Interface added
onos@root >
onos@root >
onos@root >
onos@root > interfaces
h1: port=of:0000000000000002/2
h2: port=of:0000000000000003/1
h3: port=of:0000000000000003/2
h4: port=of:0000000000000003/3
h5: port=of:0000000000000004/1
h6: port=of:0000000000000004/2
h7: port=of:0000000000000004/3
sta1: port=of:1000000000000001/1
onos@root >
```

Figura 58: Creación de interfaces VPLS

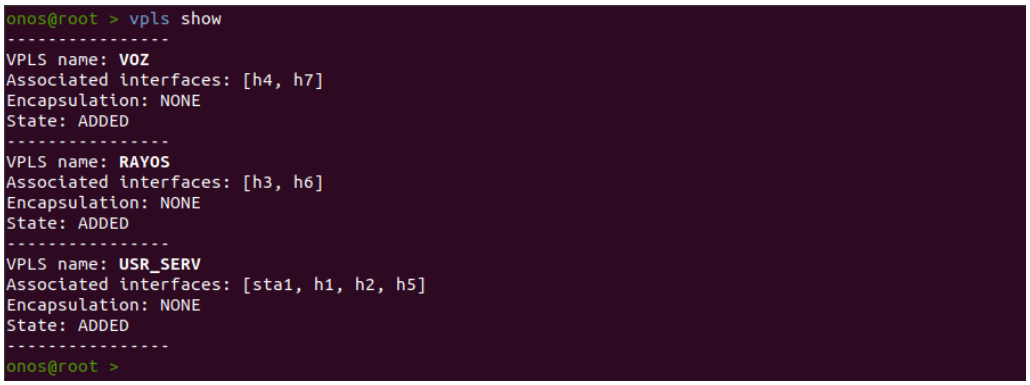


Figura 59: Resumen de configuración de VPLS

Con la configuración VPLS finalizada, si se lanza una nueva prueba de conectividad desde Mininet se podrá comprobar que el funcionamiento es el esperado.

Figura 60: Prueba de conectividad satisfactoria VPLS

Efectivamente los resultados de las pruebas obtenidos son satisfactorios. La figura anterior muestra que existe comunicación entre los host h1, h2, h5 y sta1 que pertenecen a la VPLS USR_SERV, entre los host h3 y h6 que pertenecen a la VPLS RAYOS y entre los host h4 y h7 que pertenecen a la VPLS VOZ.

Otra prueba del correcto funcionamiento es la creación de los ***intents*** en el controlador ONOS, como se muestra en la Figura 60.

Por consiguiente, teniendo en cuentas los resultados obtenidos en esta simulación se puede concluir que se ha conseguido el objetivo propuesto inicialmente. Se ha habilitado la comunicación entre los hosts pertenecientes a una misma VPLS y aunque en este apartado el diseño hacía referencia a los hosts de un único hospital, bastaría con crear las interfaces y asociar los hosts de otro hospital a las VPLS ya creadas para habilitar la comunicación.

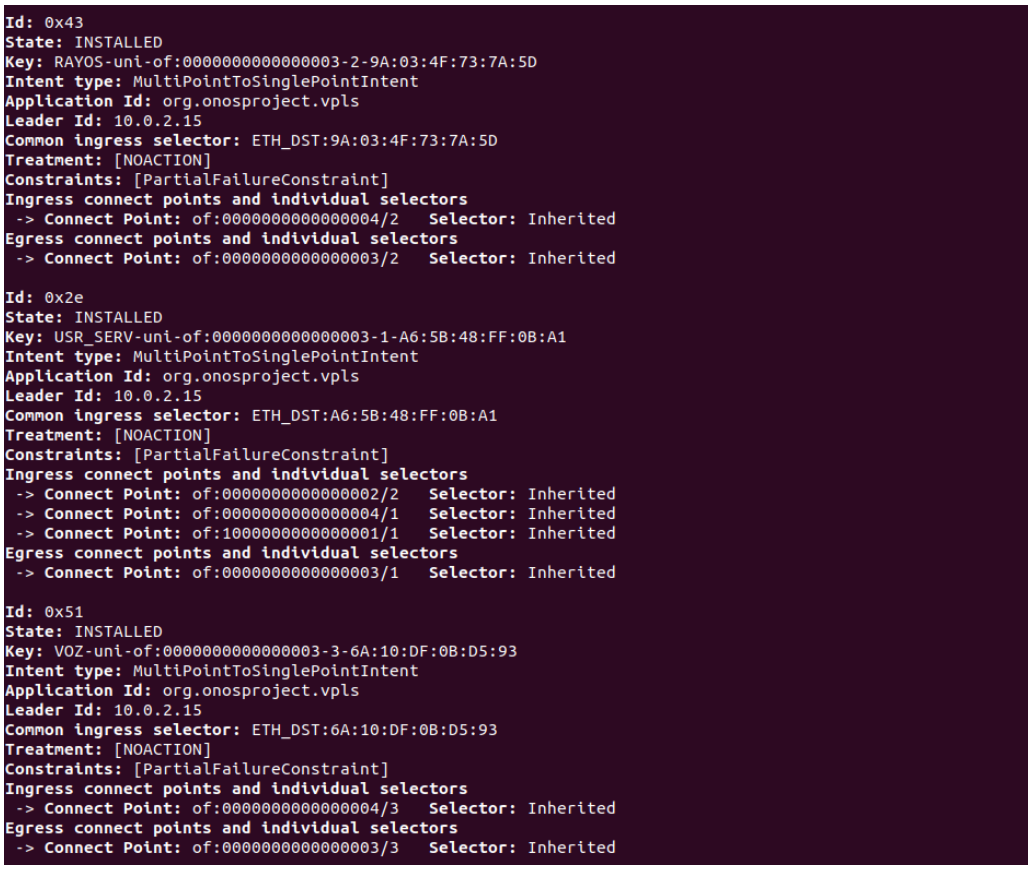


Figura 61: Intents creados en ONOS tras implementación de VPLS

5.6 Diseño de red con clúster de controlador ONOS

En este apartado del capítulo 5 se pretende abordar el diseño de red para cumplir con el objetivo de implementar un clúster del controlador ONOS para formar un sistema distribuido unificado.

Como ya se ha comentado anteriormente, se desea implementar el clúster para dotar al diseño propuesto de alta disponibilidad y redundancia ante fallos.

Para formar el clúster basta con seguir los pasos que se indican en la wiki de ONOS [9].

En este caso se empleará el clúster de ONOS con tres controladores de ONOS que ofrece la máquina virtualizada de Basic ONOS Tutorial y que también puede encontrarse en la wiki de ONOS.

En la siguiente figura se muestra el diseño a implementar.

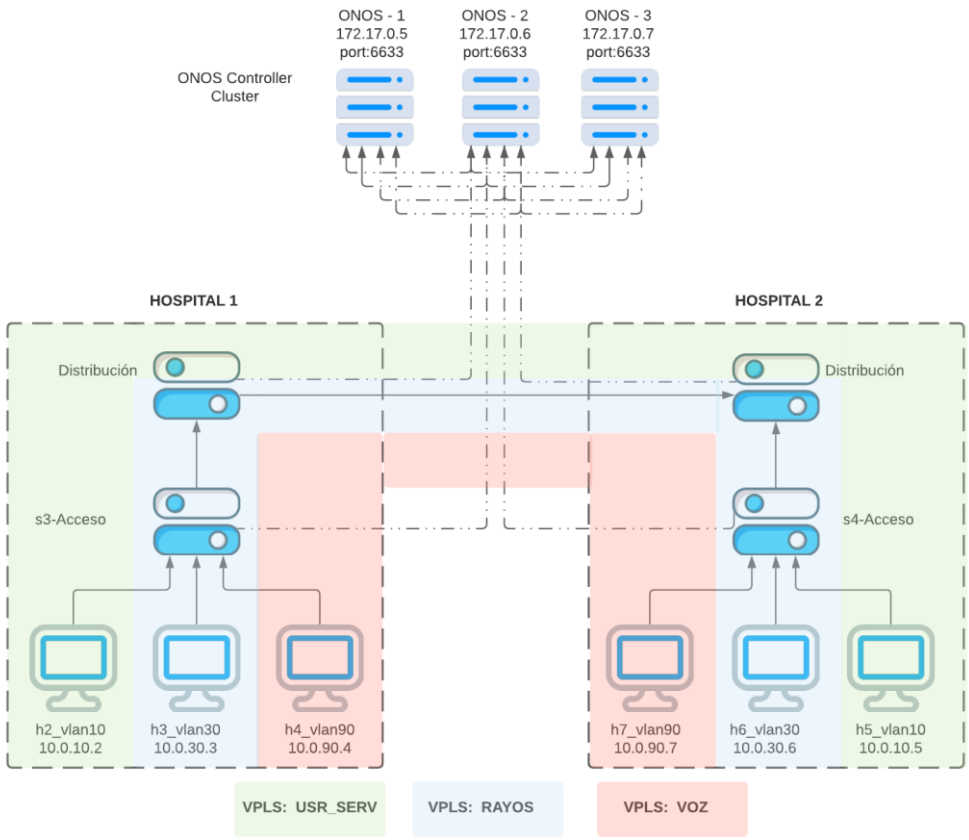


Figura 62: Diseño de red con Clúster de ONOS

Como se puede observar, el diseño consta de los siguientes elementos que entrarán en acción en la simulación:

- Tres controladores de ONOS
- 4 Switches compatibles con OpenFlow
- 6 hosts divididos en tres VLAN's

En la siguiente tabla se adjunta la información referente a los hosts:

Host	Dirección ip	VLAN	VLAN name
h2	10.0.10.2	10	Usuarios
h3	10.0.30.3	30	Radiología
h4	10.0.90.4	90	Voz
h5	10.0.10.5	10	Usuarios
h6	10.0.30.6	30	Radiología
h7	10.0.90.7	90	Voz

Tabla 6: Configuración hosts diseño de red Clúster

A continuación, se realiza la adaptación de la topología propuesta para su implementación en MiniEdit y posterior simulación.

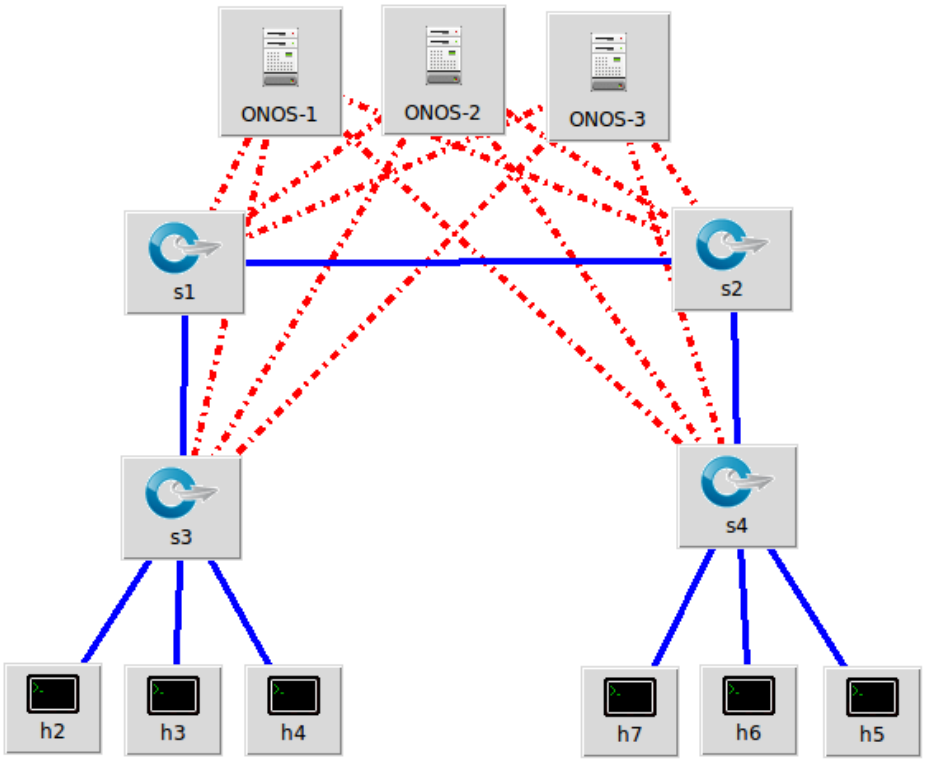


Figura 63: Miniedit: Diseño de red para Clúster de ONOS

El código resultante del diseño se puede consultar en el anexo B.

En la siguiente figura se muestra la formación del Clúster con información acerca de los dispositivos que lo conforman.

```
onos> nodes
id=172.17.0.5, address=172.17.0.5:9876, state=READY, version=1.15.0, updated=28m51s ago *
id=172.17.0.6, address=172.17.0.6:9876, state=READY, version=1.15.0, updated=28m52s ago
id=172.17.0.7, address=172.17.0.7:9876, state=READY, version=1.15.0, updated=28m52s ago
onos>
```

Figura 64: Formación del Clúster

Se verifica en las siguientes figuras que la carga en el controlador ONOS con los hosts y devices registrados, y los tres controladores operativos.

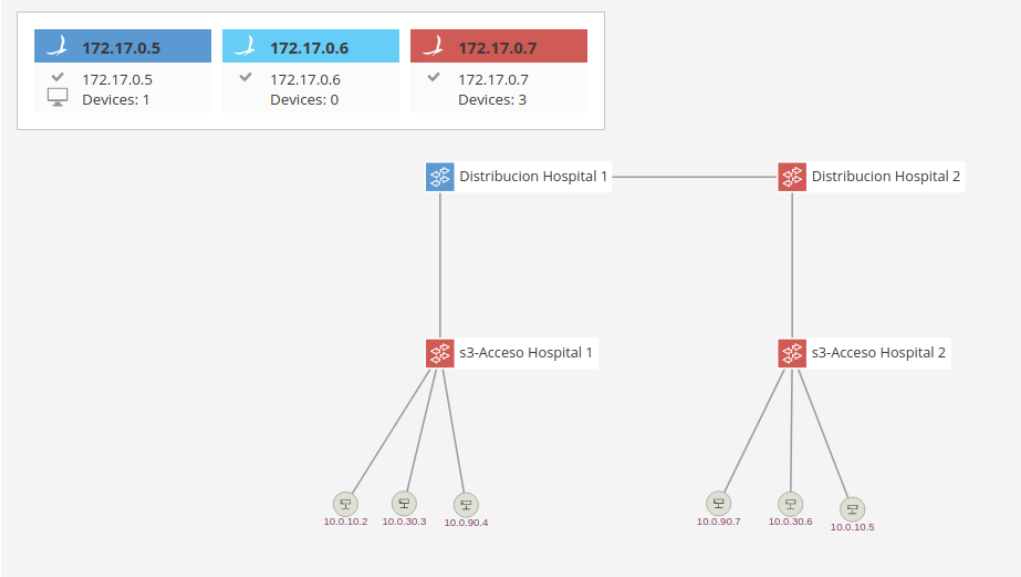


Figura 65: Carga satisfactoria del diseño con Clúster de ONOS

Se aprecia en la Figura 64, que el controlador 172.17.0.7 tiene asociados tres dispositivos, el controlador 172.17.0.6 no tiene asociados ningún dispositivo y el controlador 172.17.0.5 tiene tan sólo uno.

No obstante, se puede aplicar balanceo de carga para igualar los controladores con el comando `balance-masters`

Si aplicamos el comando, podemos comprobar en la Figura 66 que ahora el controlador 172.17.0.7 tiene asociados 2 dispositivos y los controladores 172.17.0.6 y 172.17.0.5 tienen asociado un dispositivo. De esta manera se iguala la carga entre controladores.

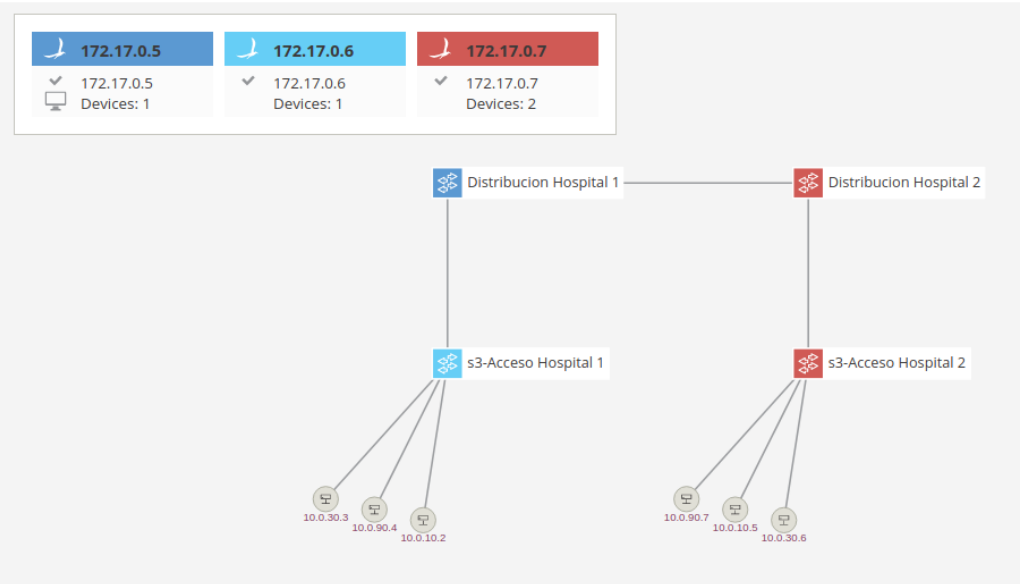


Figura 66: Aplicación de Balanceo de carga en clúster

Ahora se realizará una prueba de redundancia, para ello se introducirá el comando `shutdown` por línea de comandos en el controlador, esto provocará que el controlador que está de master se apague, forzando así que uno de los dos que están operativos coja el rol de master y se repartan los dispositivos que estaban asociados al controlador que se ha apagado.

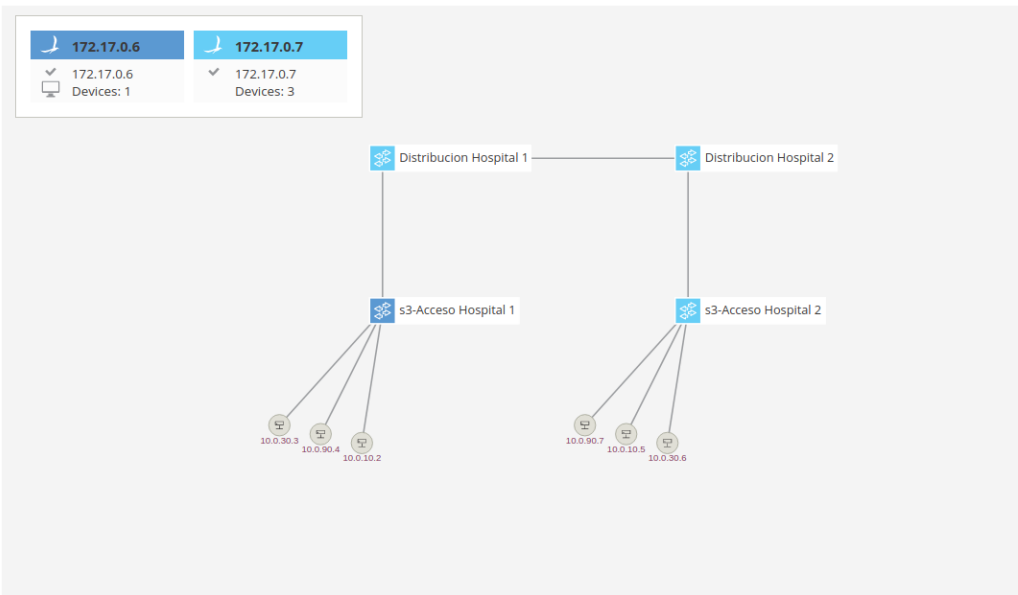


Figura 67: Comportamiento redundante Clúster

Se puede verificar por la Figura 67 que el comportamiento ha sido el esperado, ya que uno de los controladores cogió el rol de master y se repartieron los dispositivos.

Ahora se procederá a la configuración de VPLS donde se comprobará la comunicación entre hosts de la misma VLAN pero ubicados en sedes distintas.

En la siguiente tabla de expone de manera resumida la configuración de las distintas VPLS

VPLS		host	Dirección ip	VLAN
USR_SERV				
		h5	10.0.10.5	10
		h2	10.0.10.2	10
RAYOS		h3	10.0.30.3	30
		h6	10.0.30.6	30
VOZ		h7	10.0.90.7	90
		h4	10.0.90.4	90

Tabla 7: Configuración VPLS planteada para Clúster

Para la configuración de la VPLS se seguirán los mismos pasos que el apartado 5.4.

- Se crearán las interfaces asociadas a los hosts
- Se crearán las VPLS indicadas en la tabla 7
- Se asignarán los hosts a las VPLS indicadas según la tabla 7

```
onos> apps -a -s
*  21 org.onosproject.intentsynchronizer   1.15.0   Intent Synchronizer
*  22 org.onosproject.vpls                 1.15.0   VLAN L2 Broadcast Network
*  24 org.onosproject.optical-model        1.15.0   Optical Network Model
*  26 org.onosproject.openflow-base        1.15.0   OpenFlow Base Provider
*  27 org.onosproject.lldpprovider         1.15.0   LLDP Link Provider
*  28 org.onosproject.hostprovider         1.15.0   Host Location Provider
*  30 org.onosproject.drivers              1.15.0   Default Drivers
*  65 org.onosproject.openflow             1.15.0   OpenFlow Provider Suite
* 142 org.onosproject.layout               1.15.0   UI Auto-Layout
onos>
onos>
onos> hosts
id=12:B3:FF:15:9E:DE/None, mac=12:B3:FF:15:9E:DE, locations=[of:0000000000000004/2], v
lan=None, ip(s)=[10.0.90.7], gridX=65.0, gridY=550.0, locType=grid, name=-, innerVlan=
None, outerTPID=unknown, provider=of:org.onosproject.provider.host, configured=false
id=22:F8:89:10:B6:E5/None, mac=22:F8:89:10:B6:E5, locations=[of:0000000000000003/3], v
lan=None, ip(s)=[10.0.30.3], gridX=-335.0, gridY=550.0, locType=grid, name=-, innerVla
n=None, outerTPID=unknown, provider=of:org.onosproject.provider.host, configured=false
id=26:14:D3:70:52:6B/None, mac=26:14:D3:70:52:6B, locations=[of:0000000000000003/4], v
lan=None, ip(s)=[10.0.90.4], gridX=-281.0, gridY=561.6666666666666, locType=grid, name
=-, innerVlan=None, outerTPID=unknown, provider=of:org.onosproject.provider.host, conf
igured=false
id=AA:2F:77:D4:BB:FC/None, mac=AA:2F:77:D4:BB:FC, locations=[of:0000000000000004/4], v
lan=None, ip(s)=[10.0.10.5], gridX=119.0, gridY=561.6666666666666, locType=grid, name=
-, innerVlan=None, outerTPID=unknown, provider=of:org.onosproject.provider.host, confi
gured=false
id=C2:B8:80:19:95:82/None, mac=C2:B8:80:19:95:82, locations=[of:0000000000000004/3], v
lan=None, ip(s)=[10.0.30.6], gridX=173.0, gridY=573.3333333333333, locType=grid, name=
-, innerVlan=None, outerTPID=unknown, provider=of:org.onosproject.provider.host, confi
gured=false
id=EA:89:98:6F:AF:B5/None, mac=EA:89:98:6F:AF:B5, locations=[of:0000000000000003/2], v
lan=None, ip(s)=[10.0.10.2], gridX=-227.0, gridY=573.3333333333333, locType=grid, name
=-, innerVlan=None, outerTPID=unknown, provider=of:org.onosproject.provider.host, conf
igured=false
onos>
onos>
onos> interfaces
h2: port=of:0000000000000003/2
h3: port=of:0000000000000003/3
h4: port=of:0000000000000003/4
h7: port=of:0000000000000004/2
h6: port=of:0000000000000004/3
h5: port=of:0000000000000004/4
onos>
onos>
onos> vpls show
----------------
VPLS name: VOZ
Associated interfaces: [h4, h7]
Encapsulation: NONE
State: ADDED
----------------
VPLS name: RAYOS
Associated interfaces: [h3, h6]
Encapsulation: NONE
State: ADDED
----------------
VPLS name: USR_SERV
Associated interfaces: [h2, h5]
Encapsulation: NONE
State: ADDED
----------------
onos>
```

Figura 68: Configuración VPLS para diseño Clúster ONOS

Con la configuración VPLS finalizada, si se lanza una nueva prueba de conectividad desde Mininet se podrá comprobar que el funcionamiento es el esperado.

Figura 69: Prueba de conectividad satisfactoria VPLS

Efectivamente los resultados de las pruebas obtenidos son satisfactorios. La figura anterior nos muestra que existe comunicación entre los host h2 y h5 que pertenecen a la VPLS USR_SERV, entre los host h3 y h6 que pertenecen a la VPLS RAYOS y entre los host h4 y h7 que pertenecen a la VPLS VOZ.

Por consiguiente, teniendo en cuenta los resultados obtenidos en esta simulación podemos concluir que se ha conseguido el objetivo propuesto inicialmente. Se ha comprobado el correcto funcionamiento redundante del clúster de ONOS y su balanceo de carga, lo que proporciona al diseño propuesto de alta disponibilidad y redundancia ante fallos. Por otro lado, se ha habilitado la comunicación entre los hosts pertenecientes a una misma VLAN y que están ubicados en sedes distintas con la configuración de VPLS.

6. Conclusiones

Tras el estudio de las SDN para el desarrollo de este TFM, se puede decir que este nuevo paradigma en el diseño de redes, aunque más que nuevo se diría que desconocido, aporta una multitud de ventajas y posibilidades de adaptación a las altas exigencias presentadas por las nuevas tecnologías emergentes y que ya no son capaces de soportar las redes tradicionales.

A nivel personal, opté por la temática de la tecnología SDN porque me pareció algo fascinante al estudiarla en la asignatura de Redes de Nueva generación del Máster. Ese era todo el conocimiento previo que tenía acerca de esta tecnología antes de embarcarme en este trabajo, una vez finalizado y viendo claramente la actual tendencia existente hacia un modelo más centralizado y flexible, puedo asegurar que es de gran valor todo el conocimiento que he adquirido durante el desarrollo de mismo.

En cuanto a los objetivos planteados inicialmente considero que se han cumplido en términos globales. Recordemos que se han llevado a cabos 3 simulaciones o casos de estudio en los que:

- Se ha comprobado que la funcionalidad de monitorización que incorpora ONOS es una gran alternativa para cubrir la carencia que actualmente sufren los administradores de red para detectar un equipo que satura la red en la sede.
- Se ha implementado VPLS para habilitar la comunicación entre los hosts pertenecientes a una misma VLAN sin importar en qué sede se encuentran ubicados.
- Se ha comprobado el correcto funcionamiento redundante del clúster de ONOS y su balanceo de carga proporcionando así de alta disponibilidad y redundancia ante fallos al diseño.

Por otro, en cuanto a la planificación y metodología seguida a lo largo del trabajo, considero que inicialmente se definieron correctamente los diferentes hitos para un correcto avance del proyecto. No obstante, debido a problemas personales y tener que compatibilizar TFM con vida profesional y familiar, me ha resultado complicado cumplir en fecha con algunas revisiones y entregas.

Finalmente, en cuanto a líneas de trabajo futuro que no se han podido explorar en este trabajo:

- Aplicación de políticas QoS para priorización de tráfico crítico
- Adaptación del bloque WAN de los hospitales a SD-WAN

7. Glosario

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

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

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

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

OF *Openflow* es considerado uno de los primeros estándares de redes definidas por software (SDN).

OSGi *Open Services Gateway initiative* Es una tecnología de software distribuido que define una infraestructura extremadamente eficiente para el desarrollo de aplicaciones Java

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

OVS *Open vSwitch* es un software de código abierto, diseñado para ser utilizado como un switch virtual en entornos de servidores virtualizados.

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

VPLS *Virtual Private LAN Service* es una forma de proporcionar comunicación multipunto a multipunto basada en Ethernet a través de redes IP. Permite que sitios geográficamente dispersos compartan un dominio de transmisión Ethernet-

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

8. Bibliografía

[1] M. Á. Barrera-Pérez, N. Y. Serrato-Losada, E. Rojas-Sánchez y G. Mancilla-Gaona, «State of the art in software defined networking (SDN),» [En línea]. Available: https://doi.org/10.14483/22484728.14424. [Último acceso: Noviembre 2020].

[2] Mininet, «github.com/mininet/mininet/wiki/,» [En línea]. Available: https://github.com/mininet/mininet/wiki/Introduction-to-Mininet#what. [Último acceso: noviembre 2020].

[3] O. N. Foundation, « "OpenFlow Switch Specification",» [En línea] Available: https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf. [Último acceso: Noviembre 2020].

[4] NETCONF PROTOCOL. [En línea]. Available: https://es.wikipedia.org/wiki/NETCONF#:~:text=El%20Protocolo%20de%20Configuraci%C3%B3n%20de,rfc%3A6241%20RFC%206241.%5D. [Último acceso: Noviembre 2020].

[5] M. Emran and I. Kotuliak, "Real Time Experimentation Study of Software Defined Networking based on Openflow Protocol," 2019 2nd IEEE Middle East and North Africa COMMunications Conference (MENACOMM), Manama, Bahrain, 2019, pp. 1-7, doi: 10.1109/MENACOMM46666.2019.8988530.

[6] Dargahi, Tooska & Caponi, Alberto & Ambrosin, Moreno & Bianchi, Giuseppe & Conti, Mauro. (2017). A Survey on the Security of Stateful SDN Data Planes. IEEE Communications Surveys & Tutorials. PP. 10.1109/COMST.2017.2689819.

[7] OVSDB, [En línea]. Available: https://docs.openvswitch.org/en/latest/ref/ovsdb.7/ [Último acceso: Noviembre 2020].

[8] OVS, [En línea]. Available: https://docs.openvswitch.org [Último acceso: Noviembre 2020].

[9] ONOS , [En línea]. Available: https://wiki.onosproject.org/display/ONOS/System+Components [Último acceso: Noviembre 2020].

[10] Mininet-wifi, «mininet-wifi.github.io/» [En línea]. Available: https://mininet-wifi.github.io/. [Último acceso: Noviembre 2020].

[11] Ramon dos Reis Fontes and Christian Rodolfo Esteve Rothenberg, “ Wireless Network Emulation with Mininet-WiFi ”

[12] Goswami, Bhargavi. (2017). Software Defined Network, Controller Comparison.

[13] Fortinet FG-1800F [En línea]. Available: https://www.fortinet.com/content/dam/fortinet/assets/data-sheets/fortigate-1800f-series.pdf [Último acceso: Diciembre 2020]

[14] Cisco Catalyst 2960-X Series Switches [En línea]. Available: https://www.cisco.com/c/en/us/products/switches/catalyst-2960-x-series-switches/index.html [Último acceso: Diciembre 2020]

[15] OpenFlow Cisco Catalyst 2960-X Series Switches [En línea]. Available: https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst2960x/software/15-2_7_e/b_1527e_consolidated_2960x_cg/openflow.html [Último acceso: Diciembre 2020]

[16] Cisco Catalyst 4500-X Series Switches [En línea]. Available: https://www.cisco.com/c/en/us/products/collateral/switches/catalyst-4500-x-series-switches/data_sheet_c78-696791.html [Último acceso: Diciembre 2020]

[17] Cisco Plug-in for OpenFlow Configuration Guide for Catalyst 4500 Series Switches [En línea]. Available: https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst4500/XE3-7-0E/15-23E/configuration/guide/b-openflow-config/b-openflow-config_chapter_00.html [Último acceso: Diciembre 2020]

[18] Cisco Catalyst 9200 Series Switches [En línea]. Available: https://www.cisco.com/c/en/us/products/collateral/switches/catalyst-9200-series-switches/nb-06-cat9200-ser-data-sheet-cte-en.html [Último acceso: Diciembre 2020]

[19] Programmability Configuration Guide, Cisco IOS XE Amsterdam 17.3.x [En línea]. Available: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/173/b_173_programmability_cg/openflow.html [Último acceso: Diciembre 2020]

[20] Mininet [En línea]. Available: http://mininet.org/ [Último acceso: Diciembre 2020]

[21] Mininet – ONF [En línea]. Available: https://opennetworking.org/mininet/ [Último acceso: Diciembre 2020]

[22] Ignacio Campos Marcé, « Diseño de red de una organización sanitaria» [En línea]. Available: http://hdl.handle.net/10609/106829 [Último acceso: Noviembre 2020].

[23] Módulo 4 Asignatura de Redes de nueva generación. 'La virtualización de las redes' . Autor: Víctor Huertas García

[24] NOX/POX Controller [En línea]. Available: https://www.oreilly.com/library/view/software-defined-networking-with/9781783984282/b0f4c4c4-dab7-4361-ad34-28e227ed8f15.xhtml [Último acceso: Diciembre 2020]

[25] Beacon Controller [En línea]. Available: https://openflow.stanford.edu/display/Beacon/Home [Último acceso: Diciembre 2020]

[26] OpenDaylight (ODL) [En línea]. Available: https://www.opendaylight.org/what-we-do/odl-platform-overview [Último acceso: Diciembre 2020]

[27] WMware NSX [En línea]. Available: https://www.vmware.com/es/products/nsx.html [Último acceso: Diciembre 2020]

[28] Huawei Agile Controller [En línea]. Available: https://support.huawei.com/hedex/hdx.do?docid=EDOC1100062239&lang=en&idPath=24030814%7C250382819%7C250382820%7C22318457%7C21085964 [Último acceso: Diciembre 2020]

[29] REST Protocol [En línea]. Available: https://openwebinars.net/blog/que-es-rest-conoce-su-potencia/ [Último acceso: Diciembre 2020]

9. Anexos

9.1 Anexo A

En este anexo se incluye la script de configuración en Python del diseño de red planteado para el apartado 5.4 y apartado 5.5 del TFM.

```
#!/usr/bin/python

from mininet.node import RemoteController, OVSKernelSwitch, Host
from mininet.log import setLogLevel, info
from mn_wifi.net import Mininet_wifi
from mn_wifi.node import Station, OVSKernelAP
from mn_wifi.cli import CLI
from mn_wifi.link import wmediumd
from mn_wifi.wmediumdConnector import interference
from subprocess import call


def myNetwork():

    net = Mininet_wifi(topo=None,
                       build=False,
                       link=wmediumd,
                       wmediumd_mode=interference,
                       ipBase='10.0.0.0/8')

    info( '*** Adding controller\n' )
    ONOS = net.addController(name='ONOS',
                             controller=RemoteController,
                             ip='10.0.2.15',
                             protocol='tcp',
                             port=6633)

    info( '*** Add switches/APs\n')
    s3 = net.addSwitch('s3', cls=OVSKernelSwitch)
    ap1 = net.addAccessPoint('ap1', cls=OVSKernelAP, ssid='ap1-ssid',
                             channel='1', mode='g', position='1018.0,420.0,0')
    s1 = net.addSwitch('s1', cls=OVSKernelSwitch)
    s2 = net.addSwitch('s2', cls=OVSKernelSwitch)
    s4 = net.addSwitch('s4', cls=OVSKernelSwitch)

    info( '*** Add hosts/stations\n')
    h5 = net.addHost('h5', cls=Host, ip='10.0.10.5', defaultRoute=None)
    h2 = net.addHost('h2', cls=Host, ip='10.0.10.2', defaultRoute=None)
    h1 = net.addHost('h1', cls=Host, ip='10.0.210.1', defaultRoute=None)
    h6 = net.addHost('h6', cls=Host, ip='10.0.30.6', defaultRoute=None)
    sta1 = net.addStation('sta1', ip='10.0.0.1',
```

```
                    position='1090.0,567.0,0')
    h7 = net.addHost('h7', cls=Host, ip='10.0.90.7', defaultRoute=None)
    h3 = net.addHost('h3', cls=Host, ip='10.0.30.3', defaultRoute=None)
        h4 = net.addHost('h4', cls=Host, ip='10.0.90.4',
defaultRoute=None)

    info("*** Configuring Propagation Model\n")
    net.setPropagationModel(model="logDistance", exp=3)

    info("*** Configuring wifi nodes\n")
    net.configureWifiNodes()

    info( '*** Add links\n')
    net.addLink(h2, s3)
    net.addLink(h3, s3)
    net.addLink(h4, s3)
    net.addLink(h5, s4)
    net.addLink(h6, s4)
    net.addLink(h7, s4)
    net.addLink(ap1, s4)
    net.addLink(s3, s1)
    net.addLink(s4, s1)
    net.addLink(s1, s2)
    net.addLink(s2, h1)

    net.plotGraph(max_x=1000, max_y=1000)

    info( '*** Starting network\n')
    net.build()
    info( '*** Starting controllers\n')
    for controller in net.controllers:
        controller.start()

    info( '*** Starting switches/APs\n')
    net.get('s3').start([ONOS])
    net.get('ap1').start([ONOS])
    net.get('s1').start([ONOS])
    net.get('s2').start([ONOS])
    net.get('s4').start([ONOS])

    info( '*** Post configure nodes\n')

    CLI(net)
    net.stop()

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

9.2 Anexo B

En este anexo se incluye la script de configuración en Python del diseño de red planteado para el apartado 5.6 del TFM.

```
#!/usr/bin/python

from mininet.net import Mininet
from mininet.node import Controller, RemoteController, OVSController
from mininet.node import CPULimitedHost, Host, Node
from mininet.node import OVSKernelSwitch, UserSwitch
from mininet.node import IVSSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.link import TCLink, Intf
from subprocess import call

def myNetwork():

    net = Mininet( topo=None,
                   build=False,
                   ipBase='10.0.0.0/8')

    info( '*** Adding controller\n' )
    ONOS1=net.addController(name='ONOS1',
                      controller=RemoteController,
                      ip='172.17.0.5',
                      protocol='tcp',
                      port=6633)

    ONOS3=net.addController(name='ONOS3',
                      controller=RemoteController,
                      ip='172.17.0.7',
                      protocol='tcp',
                      port=6633)

    ONOS2=net.addController(name='ONOS2',
                      controller=RemoteController,
                      ip='172.17.0.6',
                      protocol='tcp',
                      port=6633)

    info( '*** Add switches\n')
    s4 = net.addSwitch('s4', cls=OVSKernelSwitch)
    s2 = net.addSwitch('s2', cls=OVSKernelSwitch)
    s1 = net.addSwitch('s1', cls=OVSKernelSwitch)
    s3 = net.addSwitch('s3', cls=OVSKernelSwitch)

    info( '*** Add hosts\n')
    h3 = net.addHost('h3', cls=Host, ip='10.0.30.3', defaultRoute=None)
```

```
    h4 = net.addHost('h4', cls=Host, ip='10.0.90.4', defaultRoute=None)
    h7 = net.addHost('h7', cls=Host, ip='10.0.90.7', defaultRoute=None)
    h2 = net.addHost('h2', cls=Host, ip='10.0.10.2', defaultRoute=None)
    h6 = net.addHost('h6', cls=Host, ip='10.0.30.6', defaultRoute=None)
    h5 = net.addHost('h5', cls=Host, ip='10.0.10.5', defaultRoute=None)

    info( '*** Add links\n')
    net.addLink(s3, s1)
    net.addLink(s4, s2)
    net.addLink(s1, s2)
    net.addLink(h2, s3)
    net.addLink(h3, s3)
    net.addLink(h4, s3)
    net.addLink(h7, s4)
    net.addLink(h6, s4)
    net.addLink(h5, s4)

    info( '*** Starting network\n')
    net.build()
    info( '*** Starting controllers\n')
    for controller in net.controllers:
        controller.start()

    info( '*** Starting switches\n')
    net.get('s4').start([ONOS1,ONOS2,ONOS3])
    net.get('s2').start([ONOS1,ONOS2,ONOS3])
    net.get('s1').start([ONOS1,ONOS2,ONOS3])
    net.get('s3').start([ONOS1,ONOS2,ONOS3])

    info( '*** Post configure switches and hosts\n')

    CLI(net)
    net.stop()

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