ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

Implementación de un laboratorio virtual para aprendizaje de SDN
(Virtual lab implementation for SDN learning)

Para acceder al Título de

Graduado en Ingeniería de Tecnologías de Telecomunicación

Autor: Rubén Isa Hidalgo

Octubre - 2016

E.T.S. DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACION

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por: RUBÉN ISA HIDALGO
Director del TFG:
Título: "Implementación de un laboratorio virtual para aprendizaje de redes SDN"
Title: "Virtual lab implementation for SDN learning**"**

Presentado a examen el día: 26/10/2016.

para acceder al Título de

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del Tribunal:

Presidente (Apellidos, Nombre):	Santamaría Caballero, Luis Ignacio.
Secretario (Apellidos, Nombre):	García Gutiérrez, Alberto Eloy.
Vocal (Apellidos, Nombre):	Sanz Gil, Roberto.

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente

Fdo.: El Secretario

Fdo.: El Vocal

Fdo.: El Director del TFG
(sólo si es distinto del Secretario)

Vº Bº del Subdirector

Trabajo Fin de Grado Nº
(a asignar por Secretaría)

En primer lugar, agradecer a Alberto García la oportunidad, ayuda y facilidades prestadas para finalizar éste trabajo. Has hecho que resulte muy llevadero hasta el final.

A mi familia, por entender el esfuerzo que requiere llegar hasta aquí y poder contar con su apoyo en todo momento. Porque sin vosotros no sería quien soy hoy, gracias.

Y, por último, mencionar a mis amigos. A los que llevan años aguantándome, a los que han ido apareciendo a lo largo del camino, pero sobre todo a los que siguen aún a mi lado. No lo habría conseguido sin ellos.

Índice

Índice de figuras

Capítulo 1

Introducción

1.1. Motivación y objetivos

En los últimos tiempos han surgido nuevos servicios de red que han hecho que las tecnologías que llevamos usando desde hace cincuenta años lleguen prácticamente al límite de sus capacidades. [1]La aparición en escena del 'big data', la introducción del 'cloud computing', las aplicaciones en tiempo real y el hecho de que las comunicaciones modernas no sean exclusivamente entre cliente y servidor, sino que es usual que se genera gran cantidad de tráfico máquina a máquina antes de devolver los datos al usuario final hacen que sea necesario no sólo un arreglo temporal, como ya se ha intentado creando nuevos protocolos, sino una solución global que afronte los problemas de las redes modernas y proponga métodos de comunicación más eficaces.

De esta forma nace el concepto: 'creación de redes definidas por software' (Software Defined Networking), que propone el control de los dispositivos de la red desde un software externo, apoyándose en un protocolo creado para tal fin llamado OpenFlow.

El propósito general de este proyecto es la creación de un entorno software de trabajo que permita, mediante la integración de varios elementos, la puesta en marcha y práctica con redes definidas por software (Software Defined Networks).

Se pretende utilizar el software de emulación de redes Mininet integrado en el entorno GNS3, lo que permitirá la creación y análisis de este tipo de redes de manera muy intuitiva. Además, se irá un paso más adelante en el manejo de la arquitectura SDN y se contará con el apoyo de un controlador externo a la herramienta Mininet, en algunos puntos del trabajo, llamado OpenDayLight. Servirá para tener una ayuda gráfica a la hora de crear topologías con el emulador (ya que sólo dispone, en principio, de un entorno de introducción de comandos) y monitorizar los switches por donde pasa la información, aunque sin profundizar mucho en la estructura de tramas de la arquitectura OpenFlow. Por último se propone la utilización de una interfaz gráfica de usuario que acompaña al software Mininet, llamada Miniedit, que permite la creación de topologías más complejas de una forma más didáctica e intuitiva que la

línea de comandos. Además, se tratará una última forma de crear y simular estas redes, consistente en la creación de ficheros escritos en el lenguaje de programación Python, que simplifican la tarea de tener que aprender a manejar el diccionario de comandos de la herramienta de emulación. Todo este montaje se apoyará en la utilización de dos analizadores de tráfico: 'tcpdump' y 'Wireshark', que ayudarán a ver las rutas que toman los paquetes en función de lo que se le diga al controlador SDN.

1.2. Estructura del proyecto

Tras esta breve introducción, se iniciará una revisión del estado del arte de esta tecnología para situarse en el punto en que actualmente se encuentra esta nueva arquitectura de red.

Visto esto, se procede a explicar en el segundo capítulo cada uno de los componentes que se utilizarán a lo largo de la realización del proyecto y, una vez se entienda el propósito de cada uno, se pasará en el capítulo tres a la integración de los mismos en un único entorno, comenzando por la instalación de todos los elementos y de los componentes necesarios para su funcionamiento hasta la configuración de red, que permitirán el desarrollo de los puntos explicados en el apartado anterior.

Por último, en el cuarto capítulo, partiendo de un entorno correctamente configurado, se explicará el método de creación de redes SDN, primero desde la consola de comandos que provee Mininet, haciendo uso de órdenes que permitirán ver únicamente por pantalla la estructura de la topología que se haya creado. Para facilitar esto, se contará con el controlador externo OpenDayLight en este punto, que ayudará con la visualización de la red.
A continuación se pasará a utilizar la API Miniedit para crear y salvar una topología más compleja de red, explicando las opciones de que dispone esta aplicación y utilizando nuevamente ODL, esta vez, para ver también los nodos de la red desde un controlador real y el tráfico que pasa por un determinado switch. Para terminar se probará la ejecución de un scrip en Python desde la herramienta Miniedit, que contendrá la topología anterior, simplemente para comprobar la potencia de Mininet y la cantidad de opciones que provee para la práctica con este tipo de redes.

En el punto final del capítulo cuatro se muestra la forma de lanzar, a partir de una topología en ejecución, los analizadores de tráfico que se han propuesto para capturar los paquetes que circulan por cada host de la red, sin necesidad de disponer de un controlador externo a Mininet.

El capítulo cinco incluye conclusiones acerca de lo que este entorno aporta a alguien que esté interesado en la práctica y aprendizaje de esta arquitectura de red emergente y un apartado de líneas futuras en el que se incluyen ideas para extender el proyecto, partiendo del punto en el que finaliza y profundizando más en el manejo del controlador y de las redes definidas por software.

Capítulo 2

Ámbito de trabajo

En este punto se va a hacer una revisión del estado del arte de esta tecnología, comentando los fundamentos y los elementos principales de los que consta una SDN y presentando Mininet, el simulador mediante el cual crearemos nuestros entornos, y OpenDayLight, el reciente proyecto de software libre que se ha escogido en este trabajo para desplegar la plataforma donde se alojará nuestro controlador de red.

2.1. Estado del arte

El Software Defined Networking[1] es una arquitectura de red emergente en la que se le da el control a una aplicación software llamada controlador. El término Software Defined Network (red definida por software) hace referencia a esta arquitectura en la cual se permite separar el plano de control del plano de datos con el objetivo de conseguir redes más programables, flexibles y automatizables. Esta migración del control, permite que la infraestructura subyacente sea abstraída para que aplicaciones y servicios de red puedan tratar a la red como una entidad virtual. El controlador de una SDN es una entidad centralizada, es decir, que puede tratarse de varias instancias físicas o virtuales, pero se comporta como un único componente, que mantendrá un estado global de la red (o fragmento de ésta), permitiendo que empresas y operadores ganen control sobre toda la red desde un único punto lógico, simplificando enormemente el diseño y operación.

Además, SDN simplifica también los dispositivos de red, ya que no es necesario que entiendan y procesen miles de protocolos, sino únicamente aceptar las instrucciones del controlador, definidas por el administrador de la red. Si nos fijamos en una arquitectura de red tradicional, consiste en un conjunto de medios de transmisión (aire, fibra óptica, cobre, ...) y conmutación (switches, routers, ...). La gestión de estas redes es distribuida, cada elemento de conmutación incorpora un firmware que toma sus propias decisiones en función de determinados campos en las tramas y paquetes recibidos. Si bien es verdad que han ayudado a mitigar los efectos de los requerimientos de los nuevos servicios de red, debemos darnos cuenta de que ya no resultan eficientes y presentan limitaciones, algunas de las cuales se van a detallar a continuación:[2]

- Los protocolos actuales están basados en RFC's, y cualquier modificación que

se quiera introducir habrá de pasar un largo proceso de estudio y aprobación por parte de los organismos competentes.

- No se promueve la investigación y el desarrollo debido a que los fabricantes y administradores son reacios a la hora de experimentar e introducir nuevos modelos en redes que ya funcionan, según ellos, de forma satisfactoria.
- Estas arquitecturas de red tradicional no fueron diseñadas para soportar los anchos de banda y servicios requeridos actualmente, como es el caso del streaming o de los juegos en línea.
- Presentan poca flexibilidad y son difíciles de administrar y configurar, ya que se comportan en base a los protocolos que los fabricantes incluyen en sus dispositivos. Si queremos mover un dispositivo en nuestra red, es probable que haya que modificar tanto las reglas del router, del firewall, ... por lo que los encargados de realizar estas tareas prefieren mantener una estructura estática, que ocasiona la pérdida de dinamismo en la red, lo que desemboca en una mala adaptación a los cambios de tráfico.

En una SDN la clave es la anteriormente mencionada separación del plano de control del plano de datos. Así, quedaría algo equivalente al modelo tradicional de red, pero se añadiría una capa software de control por encima compuesta por el controlador, encargado de gestionar la red. De forma conceptual, una red definida por software se puede dividir en diferentes capas lógicas, que se detallan en la Figura 2.1

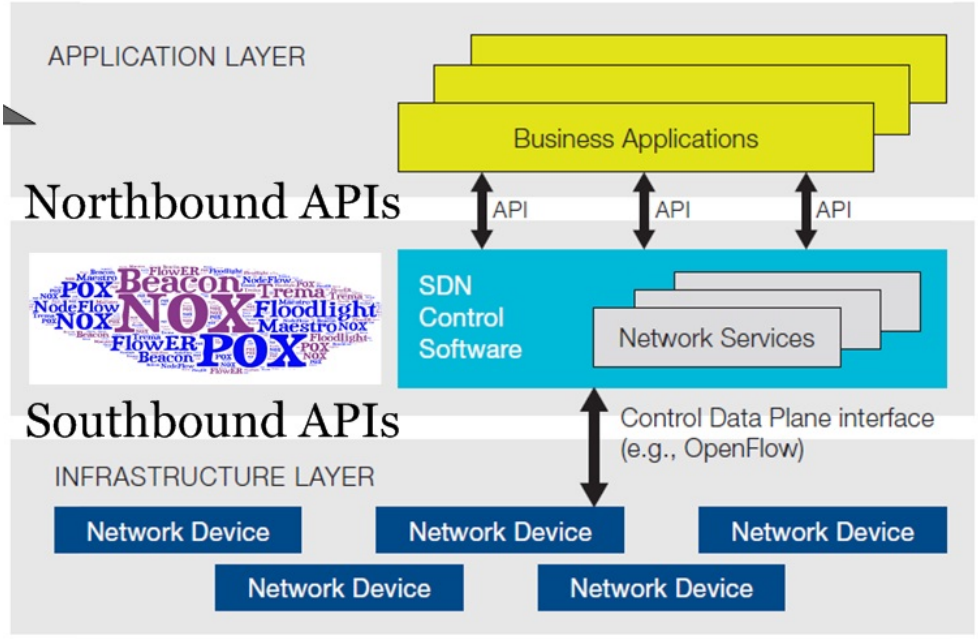


Figura 2.1: Arquitectura SDN [3]

La capa inferior está compuesta por la infraestructura de red, los hosts, switches o routers y los medios de transmisión. En la capa intermedia encontramos el controlador

SDN, que posee una visión global de la red y toma las decisiones y programa las tablas de flujo de los elementos de la capa inferior. La comunicación entre estas dos capas se realiza por medio de las conocidas como Southbound APIs y Northbound APIs. Southbound API tiene como función permitir al controlador comunicarse con los elementos de conmutación de la red y programar la lógica de las comunicaciones en el hardware. La capa superior es la capa de aplicación. Compuesta por las aplicaciones personalizadas por los usuarios, y que incorporan las Northbound APIs. Aquí es donde se produce el cambio total con respecto a las redes tradicionales. Las Northbound APIs incorporan los patrones de uso de la red de capa de aplicación, y tienen la función de comunicar esos patrones a la capa de control, donde se tomarán las decisiones pertinentes y serán comunicadas a la capa inferior mediante las Southbound APIs.

2.2. El protocolo OpenFlow

OpenFlow[4] es uno de los primeros estándares de comunicación definida entre el switch OpenFlow y el controlador en una arquitectura SDN. Facilita la programabilidad de la red mediante la configuración, gestión y control de flujos de datos desde un software centralizado. Permite particionar el tráfico, y decidir la mejor ruta y forma en que los paquetes son procesados. Está enfocado al control del tráfico, la seguridad, la creación de nuevas formas de enrutamiento, entre otras cosas. Cuando se establece una conexión cada parte envía un mensaje de HELLO[5] que contiene la versión más alta del protocolo OpenFlow que soporta. El receptor toma como versión a utilizar la más pequeña entre la recibida en el mensaje de HELLO y la que él soporta. Si es soportada se iniciará la conexión, en caso contrario se enviará de vuelta un mensaje HELLO-FAILED y la conexión terminará.
El protocolo OpenFlow[6] consta de tres tipos diferentes de mensajes:

1. Mensaje del controlador al switch: Lo inicia el controlador, y tiene la función de conocer y actuar sobre el estado del switch. Le envía una petición de características al switch y este debe responder con las capacidades disponibles. Podemos distinguir cuatro tipos dentro de este mensaje:

   Modify-State: Su objetivo principal es añadir y eliminar entradas en las tablas de flujo y fijas características de los puertos.

   Read-State: pregunta al switch por estadísticas del tráfico.

   Send-Packet:envía paquetes por un puerto específico en el switch.

   Barrier request/reply:se usa para recibir notificaciones de operaciones completadas.

2. Mensajes asíncronos: Existen cuatro tipos de mensajes que los switches envían al controlador tras la llegada de un paquete, tras un error, o tras un cambio de estado:

   Packet-in: es enviado al controlador cuando se recibe cualquier paquete que no tenga coincidencias en la tabla de flujo, o si la acción que corresponde en la tabla coincidente es reenviar el paquete al controlador.

Flow-modify: enviado cuando una entrada es añadida a la entrada de flujos de un switch.

Port-status: enviado al cambiar el estado de un puesto determinado.

Error.

3. Mensajes simétricos: Se envían sin solicitud en cualquier dirección, y se distinguen tres tipos:

   Hello: mensajes intercambiados durante la negociación de una conexión.

   Echo request/reply: pueden ser enviados por el switch o por el controlador, con el fin de comprobar la comunicación entre ellos. Debe ser respondido con un mensaje Echo replpy.

   Vendor message: está pensado para futuras revisiones de OpenFlow, proporciona una manera de ofrecer características adicionales.

Un switch OpenFlow puede tener diferentes tablas de flujos, en las cuales se realiza la búsqueda de coincidencia con los paquetes entrantes. Cada entrada en una tabla de flujos consta de tres campos principalmente:

1. Una cabecera que define el flujo.

2. Contadores para llevar la cuenta de la coincidencia de paquetes que pueden actualizarse por flujo, por tabla, puesto y por cola.

3. Acciones a realizar cuando se encuentra una coincidencia entre un paquete y una tabla de flujos.

   Una entrada en la tabla de flujo se identifica por el conjunto de los campos Match fields y Priority, y tendrá un formato como el de la Figura 2.2

   Match fields: contiene puerto y cabecera, y opcionalmente metadatos especificados en una tabla anterior.

   Priority: determina la prioridad de la entrada de flujo.

   Counters: se actualiza cuando se encuentran coincidencias.

   Instructions: acciones que serán ejecutadas si los paquetes se relacionan con alguna entrada de la tabla de flujo.

   Timeout: tiempo máximo de espera antes de que un flujo caduque.

   Cookie: es un valor usado por el controlador para modificaciones del flujo, filtrar estadísticas, ... No es usado cuando se están procesando paquetes.

Figura 2.2: Campos de los que consta la tabla de flujos en un switch OpenFlow[5]

2.3. El controlador

En la arquitectura SDN el controlador[2] es el elemento principal. Se trata del dispositivo que implementa las reglas de la red, ejecuta las instrucciones que le llegan de las aplicaciones y las distribuye entre los diferentes dispositivos de capa física de la red. Se encarga de decidir qué hacer con los paquetes que no encajan en las entradas de las tablas de flujo, además de gestionarlas. Los controladores se diferencian entre sí, fundamentalmente en cuanto al lenguaje de programación y plataforma, pero en esencia realizan las mismas funciones: comunicarse, mediante el protocolo OpenFlow, con los dispositivos de la red.

2.3.1. OpenDayLight (ODL)

OpenDayLight[7] se trata de un proyecto de código abierto (Open Source) que tiene como objetivo acelerar la difusión de la innovación en el diseño e implementación de un estándar abierto y transparente de redes definidas por software (SDN). Pretende convertirse en una plataforma abierta que utilicen todas las empresas, evitando que las aplicaciones privadas restrinjan el crecimiento del mercado, reduciendo al mismo tiempo los costes de desarrollo.

La principal ventaja de esta propuesta es que elimina barreras, ya que algunas organizaciones no quieren comprometerse con fabricantes específicos que en un futuro puedan bloquear su desarrollo. Al tratarse de una plataforma común, las empresas podrán optar por tecnologías de diversos fabricantes.

La plataforma que proporciona este proyecto (ODL) puede ser desplegada directamente sin necesidad de ningún otro componente. Esto es debido a la arquitectura de la plataforma, que proporciona un conjunto de funciones básicas para las aplicaciones y a la gran variedad de colaboradores que contribuyen en el proyecto.

2.4. Mininet

[8][9]Se trata de un emulador de red que crea redes de hosts virtuales, switches, controladores y enlaces. Se utiliza la virtualización ligera para hacer que un solo sistema parezca una red completa. Es, hasta ahora, la plataforma de pruebas de red de código abierto que proporciona apoyo a la investigación de las SDN mas conocida. Además, las aplicaciones de red basadas en Unix/Linux, también se pueden ejecutar en los hosts virtuales. En una red OpenFlow emulada por Mininet, una aplicación de controlador real OpenFlow se puede ejecutar en una máquina externa o en el mismo equipo en el que se emulan los hosts virtuales. Esta herramienta utiliza el kernel de Linux y otros recursos para emular elementos de la SDN como el controlador, los switches OpenFlow y los hosts.

En la Figura 2.3 se muestra la interfaz del emulador y se detallan algunas de las funciones básicas que permiten comenzar a manejar esta herramienta. Para profundizar en el manejo de los comandos de emulación en Mininet consultar el

Anexo A[10].

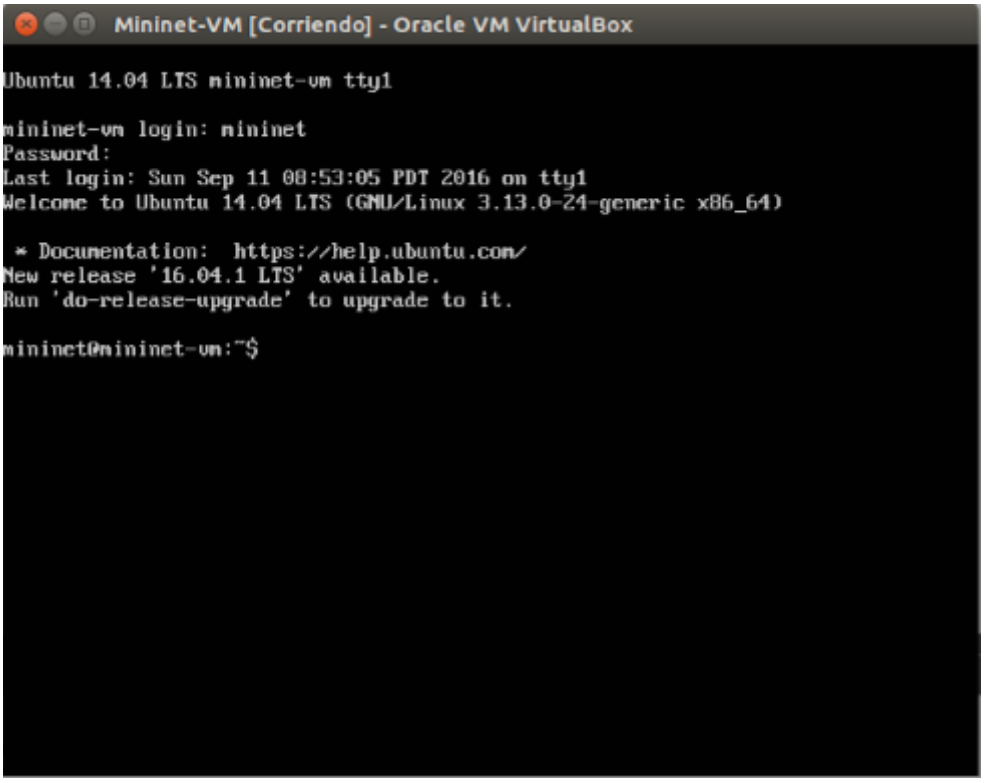


Figura 2.3: Imagen Interfaz Mininet

Se trata de una ventana de comandos donde, por medio de estos, se podrán crear distintas topologías de red y especificar los parámetros de los elementos que dentro se encuentran.

Mediante el comando:

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

Mininet inicia una topología por defecto que consta de un switch, un controlador (local) y dos hosts, como se muestra en la Figura 2.4.

Figura 2.4: Topología por defecto en Mininet

La instrucción 'sudo mn' puede venir sola, como acabamos de ver, o acompañada de diferentes parámetros, entre estos algunos de ayuda, para limpiar y reiniciar el

emulador y otros para ejecutar topologías de distintos tipos. Las topologías disponibles son: minimal, single, linear, tree y personalizadas (custom). En cualquiera de estas existe un controlador, varían en el número de hosts, switches y los enlaces entre estos.

1. Single: Se lanzará un switch conectado a N hosts.

```
mininet@mininet-vm:~$ sudo mn --topo single, N
```

2. Linear: Se trata de una topología en la que cada switch se conecta con otro de forma lineal, y cada switch tiene un host conectado.

```
mininet@mininet-vm:~$ sudo mn --topo linear, K, N
```

3. Tree: Se creará una topología en árbol con profundidad N y anchura M.

```
mininet@mininet-vm:~$ sudo mn --topo tree,
                        depth=N,fanout=M
```

4. Custom: Será necesario para este tipo de topologías crear un archivo en python con su descripción. Las topologías de este tipo se quedarán guardadas en /mininet/custom con extensión .py. Se hablará de ellas más adelante en este trabajo.

Una vez conocidas las diferentes topologías que Mininet es capaz de emular, es necesario atender a una serie de opciones con las que cuenta la herramienta a la hora de crear estas redes, que serán de utilidad a lo largo de este trabajo.

En primer lugar, Mininet utiliza un controlador por defecto incorporado por la distribución OpenFlow que actúa como un switch clásico, lo que significa que va creando una tabla donde se asocian las direcciones MAC con los puertos, según va recibiendo tramas. Lo que hace después de asociar la MAC con un puerto es enviar un mensaje OpenFlow al switch para crear una nueva entrada en su tabla de flujo. OpenFlow nombra a este tipo de controlador Ethernet Learning Switch. El fin primario de este controlador por defecto es ayudar a entender el flujo de mensajes y la separación del plano de control y datos, por lo que no admite componentes creados por el usuario. Mininet admite la utilización de controladores reales. Al iniciar una red en este emulador, cada switch se puede conectar a un controlador remoto que podría estar en la máquina virtual, fuera de ésta, en el equipo local o en cualquier parte del mundo.

En este trabajo se va a contar con un controlador externo (OpenDayLight), como se explicará más adelante debido a las ventajas que ofrece, a pesar del nivel de dificultad en el manejo, sobre el controlador por defecto de Mininet. A pesar de esto, cuando se quiera conectar Mininet a una red mayor y debido a la limitación en la utilización de interfaces que impone VirtualBox, se utilizará este controlador por defecto para que la red pueda funcionar.

Capítulo 3

Integración de los elementos

Una vez hecha la revisión del estado del arte de la tecnología SDN y presentados todos los elementos con los que se va interactuar en este trabajo, es momento de hacer una integración de todos ellos en un solo entorno. Se ha escogido la herramienta GNS3 como elemento unificador de los diferentes componentes del proyecto. El primer paso será explicar el proceso de instalación, la adición de los diferentes componentes, necesarios para la creación de las redes a la herramienta y área de trabajo donde se va a operar a partir de ahí. La realización completa del trabajo se ha desarrollado sobre un entorno Linux, concretamente sobre la distribución Ubuntu en su versión 16.04. Esta decisión ha sido motivada porque, aunque sobre un sistema operativo más convencional como podría haber sido Windows, con una interfaz bastante más intuitiva, la potencia de la línea de comandos de Linux, sumado a la activa comunidad que respalda a este SO ha permitido la resolución de los diferentes problemas que han ido apareciendo durante la realización del proyecto.

3.1. Instalación e integración de GNS3 y Mininet

GNS3[11] es un entorno gráfico que permite emular entornos de red complejos con dispositivos de diferentes fabricantes, tan sólo eligiendo los componentes que se deseen y arrastrándolos al área de trabajo, con la opción de conectarlos mediante interfaces predefinidas entre sí, y la integración de máquinas virtuales creadas externamente por otras aplicaciones. Esta forma tan fácil de crear topologías complejas hace a GNS3 una herramienta perfecta para el entrenamiento y familiarización con dispositivos de red. Para completar la instalación de GNS3 sobre una distribución Linux se deben ejecutar las siguientes secuencias de comandos en una terminal:

```
sudo add-apt-repository ppa:gns3/ppa
sudo apt-get update
sudo apt-get install gns3-gui
```

Una vez instalado y actualizado el repositorio se ejecutará:

```
sudo apt-key adv --keyserver keyserver.ubuntu.com
        --recv-keys      A2E3EF7B
sudo apt-get update
```

```
sudo apt-get install gns3-gui
```

Terminado el proceso de instalación se procede a explicar el entorno de trabajo y el método de agregación de dispositivos necesarios para la creación de topologías de red tales como los routers, que no vienen incluidos en la instalación de GNS3.

Se trata de una interfaz muy intuitiva. Como se puede ver en la Figura 3.1, existen dos barras de herramientas principalmente, la que se puede observar a la izquierda, que consta de pestañas que clasifican los diferentes componentes que se pueden agregar a la simulación, y la situada en la parte superior, que permite guardar la topología, mostrar u ocultar las interfaces, sacar la CLI de los nodos con los que se está operando, y pausar, arrancar o detener la red, principalmente. En el centro se encuentra el área de trabajo, un espacio en blanco donde se arrastrarán los componentes necesarios para trabajar. En la parte derecha de la ventana se encuentran el 'Topology Summary', que recoge los diferentes componentes desplegados y su estado, y el 'Servers Summary', que muestra cómo están funcionando los servidores que se estén usando en ese momento. Es útil, por ejemplo, para saber si se tiene suficiente RAM en un momento determinado para ejecutar alguna aplicación. Ambos menús pueden esconderse para aumentar el el tamaño del área de trabajo.

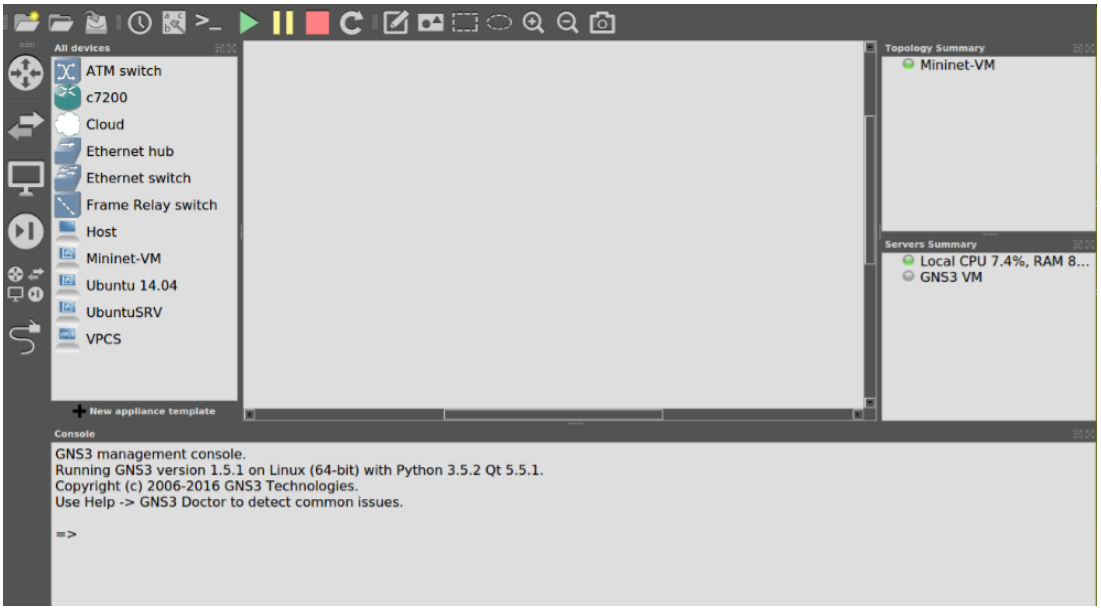


Figura 3.1: Entorno de trabajo de GNS3

La forma de añadir una nueva IOS (Internetwork Operating System) a GNS3 es desplazarse por el menú de la parte superior hasta la opción 'Edit' y seleccionar la pestaña 'Preferences'. Una vez abierto este menú, se deberá acudir al apartado 'IOS Routers' en la sección 'Dynamips' y hacer click en 'New' para añadir un nuevo modelo de router, como se muestra en la Figura 3.2. Habrá que seguir el asistente, seleccionando una imagen existente que previamente se habrá descargado, en el caso particular de este trabajo, se ha elegido la IOS de un router Cisco, modelo c7200. Una vez seleccionada la imagen se podrá pasar a configurar el nombre, la RAM por

defecto y el número y tipo de interfaces con las que contará el aparato. Una vez finalizado el último paso, ya estará disponible el router en la barra de herramientas situada a la izquierda del área de trabajo.

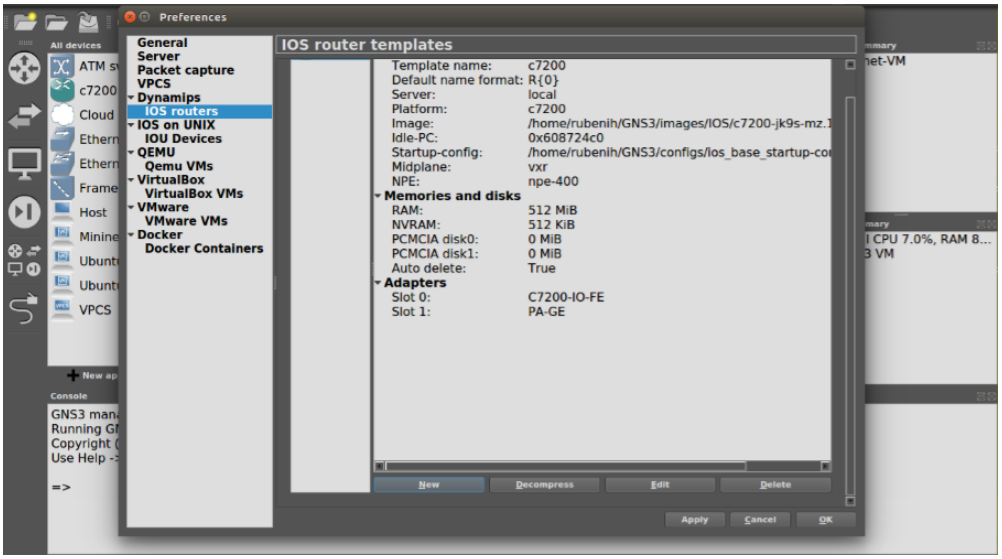


Figura 3.2: Adicionando una IOS

A continuación se detalla el proceso de instalación y posterior configuración para su integración en GNS3, de la herramienta Mininet sobre una máquina virtual creada con VirtualBox[12], un software de virtualización que permite instalar sistemas operativos 'invitados' dentro de un 'anfitrión'. El primer paso, una vez instalado VirtualBox, será crear una nueva máquina virtual (VM). Para ello, seleccionar 'New' en la barra de herramientas y seguir el asistente de instalación. En primer lugar habrá que elegir un nombre para la máquina, por ejemplo: Mininet-VM. El tipo será 'Linux' y la versión 'Ubuntu (64 o 32 bits)'. A continuación, se podrá configurar el tamaño de memoria que va a tener. Se puede dejar el valor que viene por defecto (768MB). El siguiente paso es la configuración del disco duro. La opción a escoger será 'Usar un archivo de disco duro virtual existente', y habrá que seleccionar la imagen descargada de Mininet, con formato .vmdk.

Existe otra opción para crear la VM de una forma más rápida. Se trata de, una vez creada y configurada la máquina virtual en otro anfitrión, utilizar un fichero de configuración para importarla completamente sin necesidad de asignar los parámetros explicados anteriormente. En la barra de herramientas superior, seleccionar la opción 'Importar servicio virtualizado...' de la pestaña 'Archivo', y seleccionar el archivo de configuración con la extensión .ova, si se dispusiera de uno.

En este punto ya se tiene lista para usar la VM con Mininet. Sólo hace falta completar su integración en GNS3[13]. Hace falta configurar unos adaptadores de red extra en VirtualBox. Se puede hacer desde Archivo>Preferencias, acudiendo a la pestaña 'Red'. En la pestaña 'Redes solo-anfitrión' ha de hacerse click en la opción 'Agregar red solo-anfitrión' hasta tener disponibles tres adaptadores de red:

'vboxnet0, vboxnet1, vboxnet2'.

Ahora se necesita configurar la VM con los nuevos adaptadores virtuales que se acaban de añadir a VirtualBox. Haciendo click con el botón derecho del ratón sobre la VM y seleccionando la opción 'Configuración' se tendrá acceso a la pestaña 'Red' donde se pueden configurar cuatro adaptadores. Se deberá asignar al 'Adaptador 1' la opción 'Adaptador solo-anfitrión' con el nombre 'vboxnet0'. Y repetir la operación para los adaptadores dos y tres, asignando los adaptadores solo-anfitrión uno y dos respectivamente.

Es el momento de añadir esta VM a la herramienta GNS3. Dentro de la interfaz de GNS3 acudir a: 'Editar' > 'Preferencias', y situarse en la pestaña 'VirtualBox Vms'. Se seleccionará la opción 'New' y dentro del asistente se escogerá, de entre las opciones que puedan aparecer, la que tenga el nombre de la VM creada en el paso anterior. Al finalizar el proceso aparecerá en la barra de herramientas a la izquierda del entorno de trabajo lista para trabajar con ella.

Se dispone ya de la VM con Mininet integrada en el simulador de redes GNS3, pero no dispone de acceso a internet ni tiene las interfaces que se le han dado configuradas. Para conseguir esto, desde la máquina virtual se tendrá que añadir una nueva interfaz, que se configurará como NAT. En este trabajo se le ha dado a la red NAT la Ip: 10.0.2.0/24 y se le ha dado soporte dhcp. También se le ha dado una ip a cada interfaz, de esta forma, se tendrá la siguiente asignación ip-adaptador, reflejada en el Cuadro 3.1

Adaptador 1	NAT	NatNetwork	10.0.2.0/24
Adaptador 2	Sólo-Anfitrión	vboxnet0	192.168.56.1
Adaptador 3	Sólo-Anfitrión	vboxnet1	192.168.57.1
Adaptador 4	Sólo-Anfitrión	vboxnet2	192.168.58.1

Cuadro 3.1: TABLA IP-ADAPTADOR.

Una vez hecho esto, habrá que iniciar Mininet para modificar el fichero de configuración de interfaces y añadir la nueva configuración.

Ejecutar como administrador:

```
sudo nano /etc/network/interfaces
```

Y en el fichero se añadirá la información mostrada en la Figura 3.3

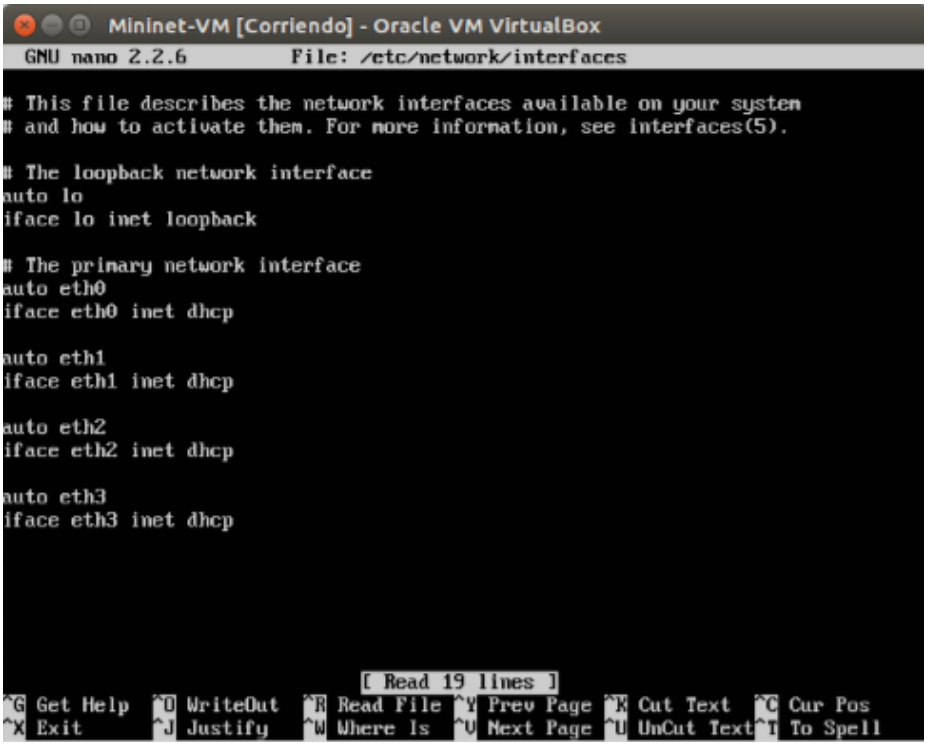


Figura 3.3: Captura fichero interfaces

Se podrá comprobar que la correspondencia interfaz-ip es la mostrada en el Cuadro 3.2

eth0	192.168.56.3
eth1	192.168.57.3
eth2	192.168.58.3
eth3	10.0.2.15

Cuadro 3.2: Se muestra la correspondencia entre las interfaces y las direcciones IP

Y al hacer un ping hacia internet, Figura 3.4, se verifica que existe acceso.

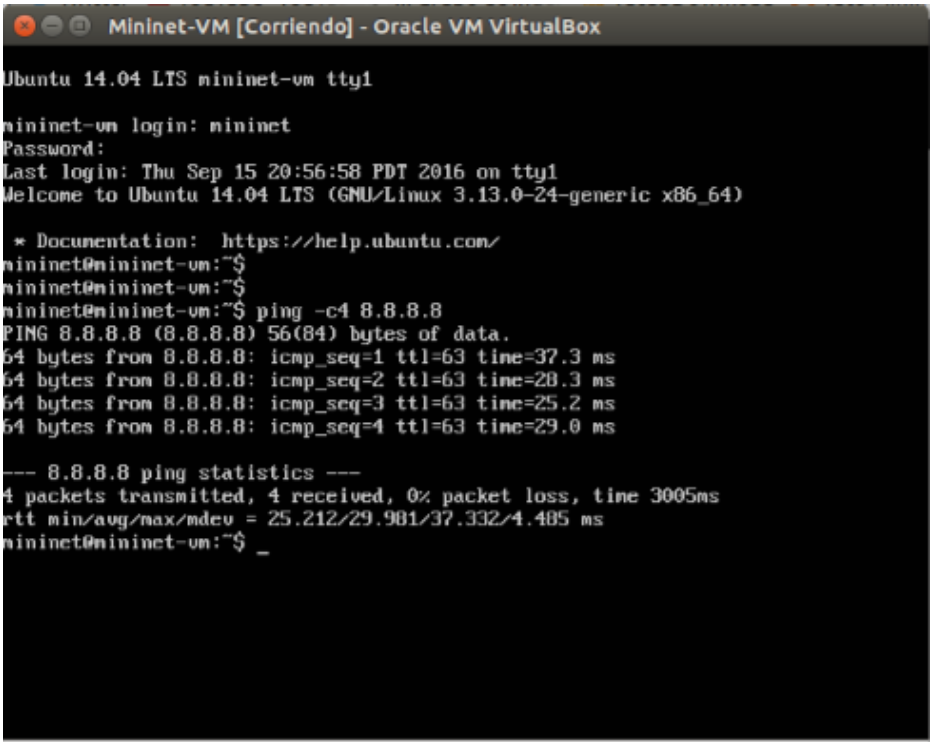


Figura 3.4: Ping realizado a la dirección 8.8.8.8 (Google.com)

3.1.1. Acceso a Miniedit y entorno de trabajo

Es una interfaz gráfica de usuario (GUI)[14] que se agregó desde la versión 2.2.0.1 de Mininet, está desarrollada en python, e incorpora los elementos básicos de una red, como hosts, switches, switches OpenFlow, controlador, routers y links principalmente, permitiendo al usuario poder crear la topología escogiendo los dispositivos que desee y que permite guardarla para seguir trabajando con ella posteriormente y exportarla en un fichero en python que podrá ser ejecutado cuantas veces se requiera.

Tiene una interfaz simple con una barra de herramientas a la izquierda de la ventana, y una barra de menú en la parte superior. Los iconos de la barra de heramientas representan las siguentes herramientas:

La herramienta de selección, Figura 3.5, es usada para mover los nodos por el área de trabajo, haciendo click con el botón izquierdo del ratón y arrastrando cualquier nodo o enlace existente hacia la posición deseada. Para seleccionar el menú de configuración de los objetos se hará click con el botón derecho del ratón y se seleccionará una de las opciones que se muestren. Para eliminar un objeto de la topología bastará con posicionar el ratón sobre este, y presionar la tecla Supr.

Figura 3.5: Herramienta de selección de Miniedit

La herramienta Host, Figura 3.6, crea nodos en el área de trabajo que llevarán a cabo la función de ordenadores (hosts). Seleccionando esta opción en la barra de herramientad y haciendo click en cualquier parte del area de trabajo aparecerán los hosts. Se pueden seguir añadiendo nodos mientras esta opción siga marcada en la barra de la izquierda. Se pueden configurar estos nodos haciendo click con el botón derecho sobre ellos y seleccionando la opción 'Propiedades' del menú.

Figura 3.6: Host en Miniedit

La herramienta Switch, Figura 3.7, crea switches OpenFlow en el área de trabajo. Estos switches deben ir conectados al controlador. Esta herramienta opera exactamente igual que la anterior. El usuario puede configurar los switches OpenFlow haciendo click con el botón derecho del ratón y eligiendo la opción 'Propiedades'.

Figura 3.7: Switch OpenFlow en Miniedit

La herramienta Legacy Switch, Figura 3.8 crea switches capaces de operar independientemente de un controlador. Estos switches no pueden ser configurados, y no se recomienda conectarlos provocando bucles, pues no tienen activado ningún algoritmo de Spanning Tree.

Figura 3.8: Legacy Switch en Miniedit

La herramienta Legacy Router, Figura 3.9 crea routers básicos que operarán independientemente de un controlador de la misma forma que los anteriores objetos. Tampoco pueden ser configurados desde la interfaz gráfica de Miniedit.

Figura 3.9: Legacy Router en Miniedit

La herramienta de NetLink, Figura 3.10, crea enlaces entre nodos colocados en el área de trabajo. El enlace se crea seleccionando uno de los nodos ya creados y arrastrando el ratón sobre el que se quiere unir a él. El usuario puede configurar las propiedades de cada enlace seleccionando la opción 'Propiedades' que aparece tras hacer click con el botón derecho del ratón sobre el enlace.

Figura 3.10: Herramienta de creación de enlaces de Miniedit

La herramienta Controller, Figura 3.11, crea un controlador. Se pueden añadir múltiples controladores a la topología. Por defecto, Miniedit crea un controlador OpenFlow 'reference controller' que implementa el comportamiento del 'Legacy Switch', aunque se pueden configurar otros tipos de controladores. El usuario puede configurar las propiedades de cada controlador de la misma forma que se han configurado todos los elementos anteriores.

Figura 3.11: Controlador en Miniedit.

La opción 'Run', Figura 3.12, lanza la topología propuesta en el área de trabajo.

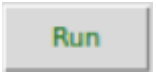


Figura 3.12: Botón 'Run' para lanzar la topología.

El botón de 'Stop', Figura 3.13, la detiene. Mientras se está ejecutando la simulación, si se hace click con el botón derecho del ratón en cualquier elemento aparecen opciones como abrir un terminal para dicho elemento, o ver las configuraciones de los switches, o seleccionar el estado de los enlaces (up o down)

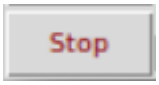


Figura 3.13: Botón 'Stop' para detener la ejecución de la simulación.

Se pueden cambiar las preferencias de Miniedit antes de comenzar a construir el escenario. En la barra de menú de la parte superior, en la opción 'Edit'(ver Figura 3.14), deberemos seleccionar 'Preferencias'.

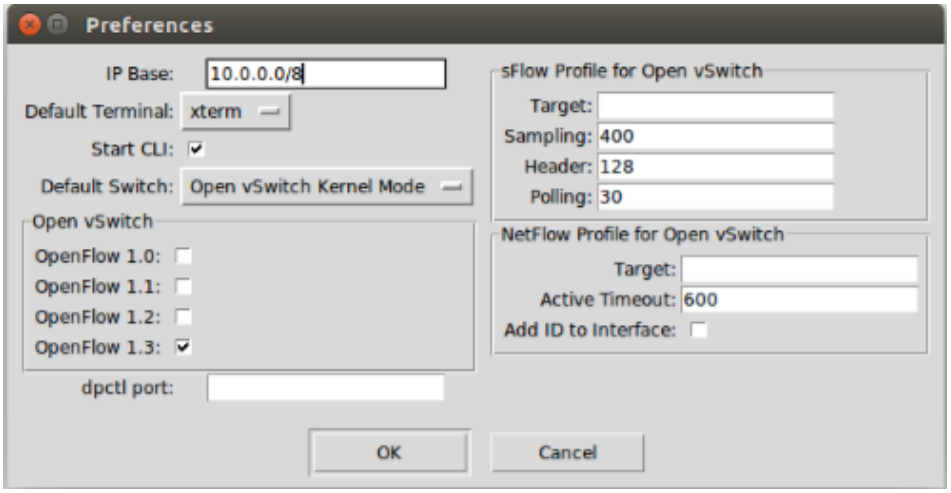


Figura 3.14: Cuadro de configuración de preferencias de Miniedit

En el cuadro de la Figura 3.14 se podrán configurar opciones como el rango de direcciones ip de los nodos y la versión de OpenFlow que soportan, que si no se especifica otra, será la 1.0.

Por defecto Miniedit no permite al usuario acceso a la interfaz de línea de comandos de Mininet. Si se quiere tener acceso a la CLI mientras una simulación está en marcha habrá que marcar la opción 'Start CLI' en el cuadro de preferencias.

A la hora de guardar la topología habrá que dirigirse a la opción 'File' en el menú de la parte superior y seleccionar 'Save' en el menú desplegable. Esto creará un archivo .mn que, para seguir una organización, habrá de ser guardado en el directorio /mininet/custom. Este archivo permitirá poder editar el escenario cuantas veces sea necesario en Miniedit. Para guardar el fichero en python que permitirá ejecutarlo desde la línea de comandos de Mininet se deberá seleccionar la opción 'Save Level 2 Script' del mismo menú desplegable y poner un nombre al archivo, que se guardará en el directorio anteriormente mencionado.

3.2. Instalación de OpenDayLight en un Servidor Ubuntu

Se creará una nueva VM sobre la que correrá un 'Ubuntu Server' en la versión 14.04. Sobre ésta, se instalará la distribución Beryllium del controlador OpenDayLight en la versión 0.4.0.

Lo primero será configurar los adaptadores de esta nueva máquina virtual. Se utilizarán dos. Uno de ellos será configurado como NAT, y el otro como 'Adaptador Solo-anfitrión'. De esta forma se conseguirá dotar a la VM de acceso a internet para poder descargar los paquetes necesarios para la instalación del controlador OpenDayLight (ODL, en adelante). Será necesario iniciar el servidor Ubuntu para modificar el fichero de configuración de interfaces como se muestra en la Figura 3.15, añadir las siguientes líneas y reiniciarlo para que los cambios se hagan efectivos:

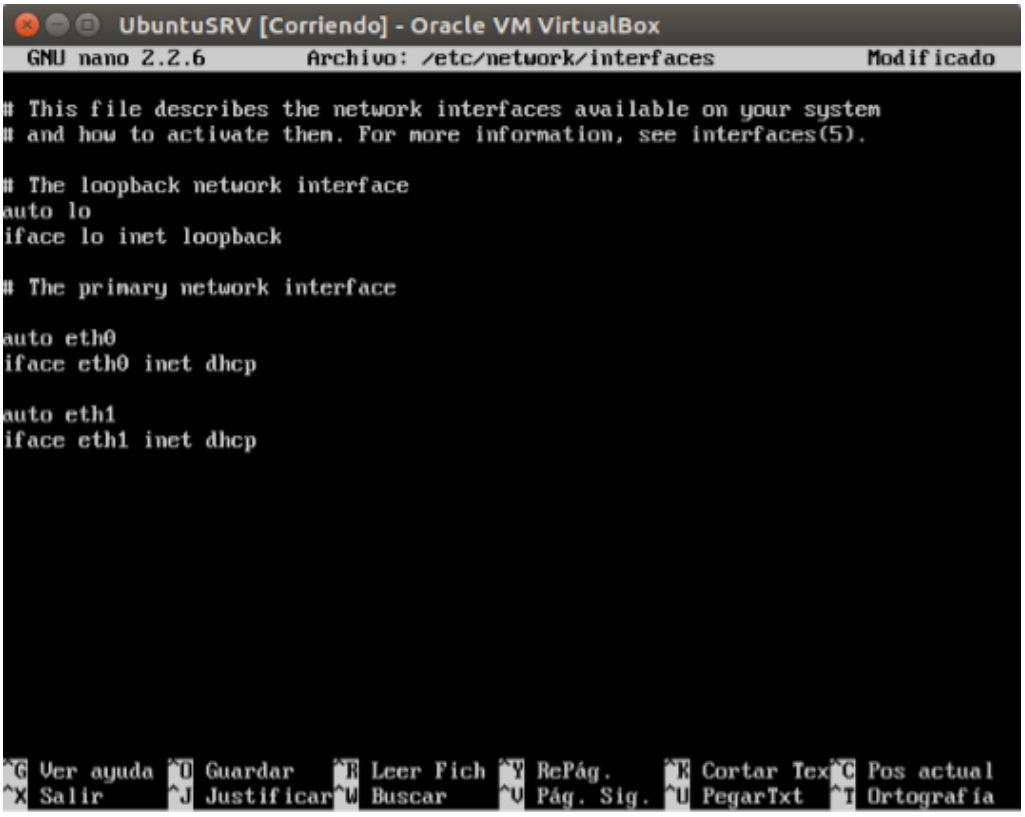


Figura 3.15: Fichero de configuración de interfaces del servidor Ubuntu.

Así, la configuración de los adaptadores e interfaces en esta VM se corresponderá con la que se muestra en el Cuadro 3.3

Adaptador 1	NAT	NatNetwork	10.0.2.15
Adaptador 2	Sólo-Anfitrión	vboxnet0	192.168.56.4

Cuadro 3.3: Se muestra la configuración de los adaptadores en el servidor Ubuntu.

El controlador OpenDayLight[15] es un programa en Java, será necesario instalar 'Java run-time' introduciendo:

```
sudo apt-get update
sudo apt-get install default-jre-headless
```

Habrá que editar el archivo 'bashrc'

```
sudo nano ~/.bashrc
```

añadiendo la siguiente línea:

```
export JAVA_HOME=/usr/lib/jvm/default-java
```

y ejecutar el archivo:

```
source ~/.bashrc
```

A continuación, introduciendo la siguiente orden, comenzará la descarga de ODL:

```
wget https://nexus.opendaylight.org/content/groups/
        public/org/opendaylight/integration/
        distribution-karaf/0.4.0-Beryllium/
        distribution-karaf-0.4.0-Beryllium.tar.gz
```

Se instalará extrayendo el archivo con extensión .tar:

```
tar -xvf distribution-karaf-0.4.0-Beryllium.tar.gz
```

Esto creará una carpeta llamada 'distribution-karaf-0.4.0-Beryllium' que contiene el software ODL. Para arrancar el controlador desde el servidor, habrá que ejecutar la siguiente orden:

```
./distribution-karaf-0.4.0-Beryllium/bin/karaf
```

y esperar a que el nuevo entorno se inicie.

Se trata de otra interfaz de introducción de comandos. Recién instalado, ODL no cuenta con ningún complemento. En este trabajo se han instalado las mínimas características que OpenDayLight debe tener para operar. La instalación de estos elementos se realiza de forma muy sencilla, mediante la introducción de un comando con formato:

feature:install [complemento]

de esta forma, se introducirá la siguiente orden para instalar estos cuatro complementos, explicados en el Cuadro 3.4:

```
feature:install odl:restconf
                odl-l2switch-switch
                odl-mdsal-apidocs
                odl-dlux-all
```

odl-restconf	permite el acceso a la API RESTCONF
odl-l2switch-switch	provee funcionalidad similar a un switch ethernet a la red
odl-mdsal-apidocs	permite el acceso a la API Yang
odl-dlux-all	es la interfaz gráfica de OpenDayLight

Cuadro 3.4: Complementos instalados en OpenDayLight.

Para obtener un listado de todos los posibles complementos que se pueden instalar se puede ejecutar:

```
feature:list
```

y si se quiere ver una lista con todos los complementos instalados:

```
feature:list -installed
```

Para detener el controlador se podrá utilizar cualquiera de estos comandos:

```
system:shutdown
logout
```

y, opcionalmente:

```
halt
```

para salir a la interfaz del servidor ubuntu sobre el que fue instalado.

3.2.1. Acceso al controlador remoto y entorno de trabajo

Con el controlador ODL instalado, además de las principales funcionalidades, y los adaptadores configurados correctamente como se ha explicado en el anterior punto, es el momento de acceder a la interfaz gráfica que proporciona esta herramienta.

El primer paso será arrancar la VM donde se ha alojado el controlador, una vez iniciada se ejecutará el comando para lanzar ODL, visto anteriormente.

Para acceder, basta con abrir el navegador web y especificar la URL de la interfaz de usuario de OpenDayLight (DLUX UI). Es necesario que esté siendo ejecutada en el momento del acceso, si no lo revocará. En el caso de este proyecto, la IP asignada es: 192.168.56.4 y el puerto por defecto definido por la aplicación el 8181.
La URL será:

http://192.168.56.4:8181/index.html

El nombre de usuario y contraseña por defecto son 'admin', como se ve en la Figura 3.16

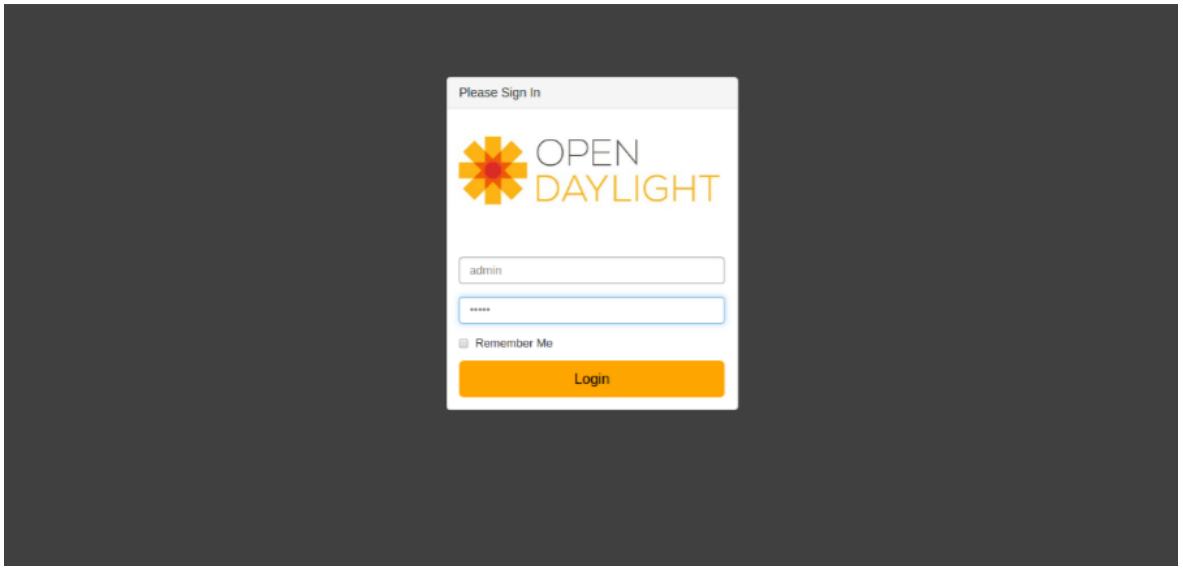


Figura 3.16: Ventana de acceso a OpenDayLight.

Una vez dentro se puede observar que se trata de una interfaz muy sencilla compuesta por un área de trabajo que ocupa la mayor parte del espacio, y un menú situado a la izquierda que consta de cuatro pestañas: Topology, Nodes, Yang UI, y Yang Visualizer.

Se puede ver la topología creada desde Mininet desde la pestaña 'Topology', Figura 3.17. Es útil si no hemos utilizado la herramienta MiniEdit y queremos ver el resultado de forma gráfica. O para comprobar los cambios que le supone a la red la creación o eliminación de enlaces entre los distintos componentes de la red.

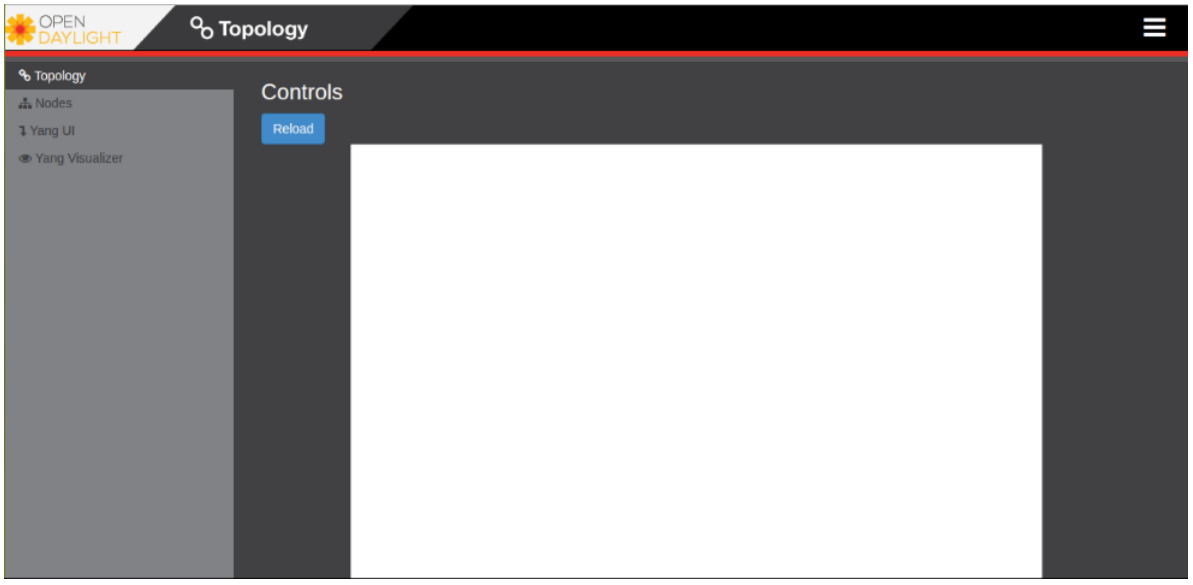


Figura 3.17: Pestaña Topology

Desde la pestaña 'Nodes', Figura 3.18 se puede ver la información acerca de cada switch en la red.

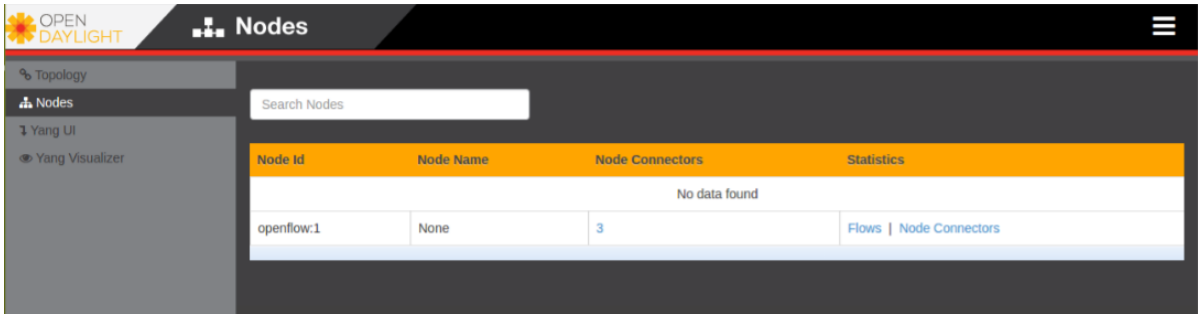
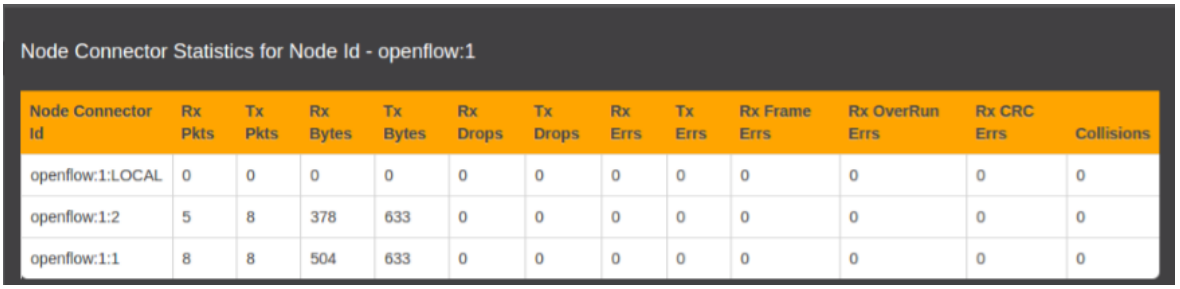


Figura 3.18: Pestaña Nodes

Haciendo click en alguna de las filas de la columna 'Node Connectors', Figura 3.19 se podrá obtener información acerca de cada puerto en ese switch:



Node Connector Id	Rx Pkts	Tx Pkts	Rx Bytes	Tx Bytes	Rx Drops	Tx Drops	Rx Errs	Tx Errs	Rx Frame Errs	Rx OverRun Errs	Rx CRC Errs	Collisions
openflow:1:LOCAL	0	0	0	0	0	0	0	0	0	0	0	0
openflow:1:2	5	8	378	633	0	0	0	0	0	0	0	0
openflow:1:1	8	8	504	633	0	0	0	0	0	0	0	0

Figura 3.19: Node connectors

Por último está la pestaña Yang UI, Figura 3.20, ya que la última de las cuatro (Yang Visualizer) no se va a utilizar. Yang es una estructura de modelado de datos similar a otras basadas en SNMP, SMI y MIB. Yang provee de una funcionalidad a los switches de la SDN de una forma análoga a como lo hace SMI para los switches que no son SDN.

Yang UI es una cliente REST gráfico para construir y mandar peticiones REST a la base de datos de ODL. Se puede utilizar para conseguir información o modificarla en esta base de datos.

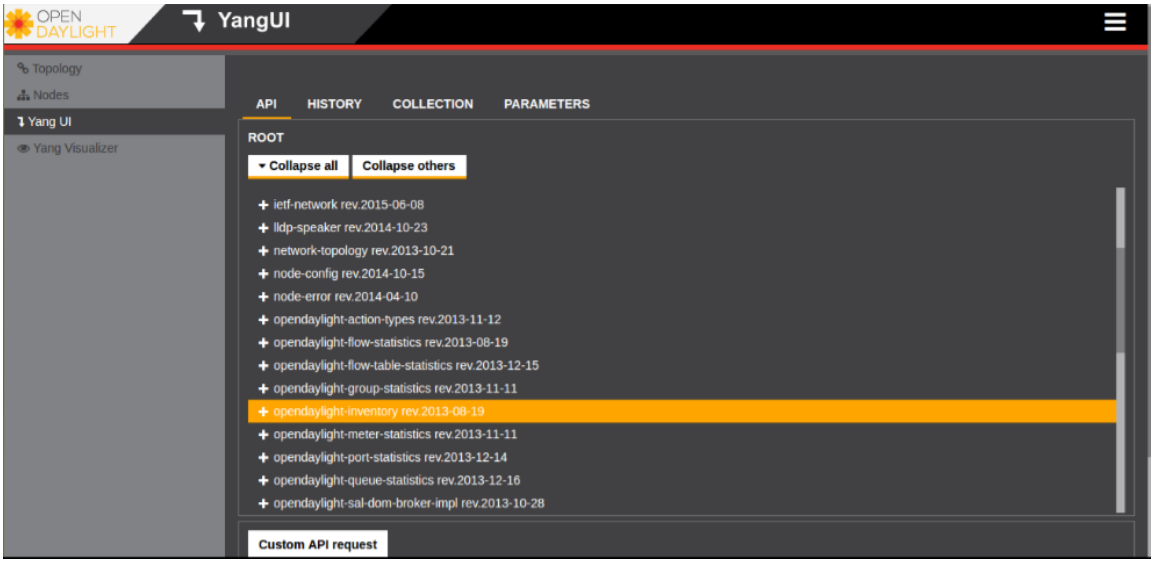


Figura 3.20: Yang UI

Para ver todas las APIs disponibles dentro de este cliente habrá que expandir todas las opciones desde el botón 'Expand all'. No todas funcionarán porque no están instaladas todas las características. Una de las que si funciona es la llamada: 'Inventory API', que se lanzará expandiendo el menú con el nombre 'inventory' desplazándose a la pestaña 'nodes' y enviando la petición GET API al controlador desde el botón 'Send'. Hacia abajo se desplegará toda la información a cerca de la red: nodos, puertos, estadísticas,... Se puede hacer click sobre los switches y las interfaces para ver los detalles de cada uno de ellos.

Capítulo 4

Implementación

Partiendo del punto en el que todas las herramientas están completamente configuradas y funcionando correctamente, se pretende comenzar con la implementación de varios ejemplos básicos para entender el funcionamiento de las mismas, así como tratar de dar una visión global de la utilidad de cada una.

Se pretende comenzar con los métodos de los que dispone el emulador Mininet para la creación de redes definidas por software: mediante la introducción de comandos por consola, utilizando la API Miniedit, o ejecutando scripts en lenguaje python que contengan la topología deseada.

Por último, se proponen dos herramientas que permitirán analizar el tráfico que circula por la red. Se explicará la forma de lanzar estos analizadores para capturar los flujos de tráfico en los puntos de interés y se comentarán los resultados.

4.1. Creando SDN con Mininet

Se propone la creación de un escenario básico en Mininet, que contará con un controlador externo (ODL) para ver la topología creada y la información que circula a través de los nodos, dos switches OpenvSwitch y dos host por cada uno de ellos. Adicionalmente, se creará una red más compleja aprovechando las facilidades que dan la ejecución de scripts en python y la API Miniedit.

4.1.1. Mediante introducción de comandos

En primer lugar habrá que arrancar GNS3. Una vez inciado, en la pestaña 'All devices' se encontrará la VM que previamente se había integrado en esta herramienta: Mininet-VM. Habrá que arrastrarla al entorno de trabajo, y una vez ahí, iniciarla, como se muestra en la Figura 4.1 . El usuario y la clave para entrar, por defecto son 'mininet' en ambos casos.

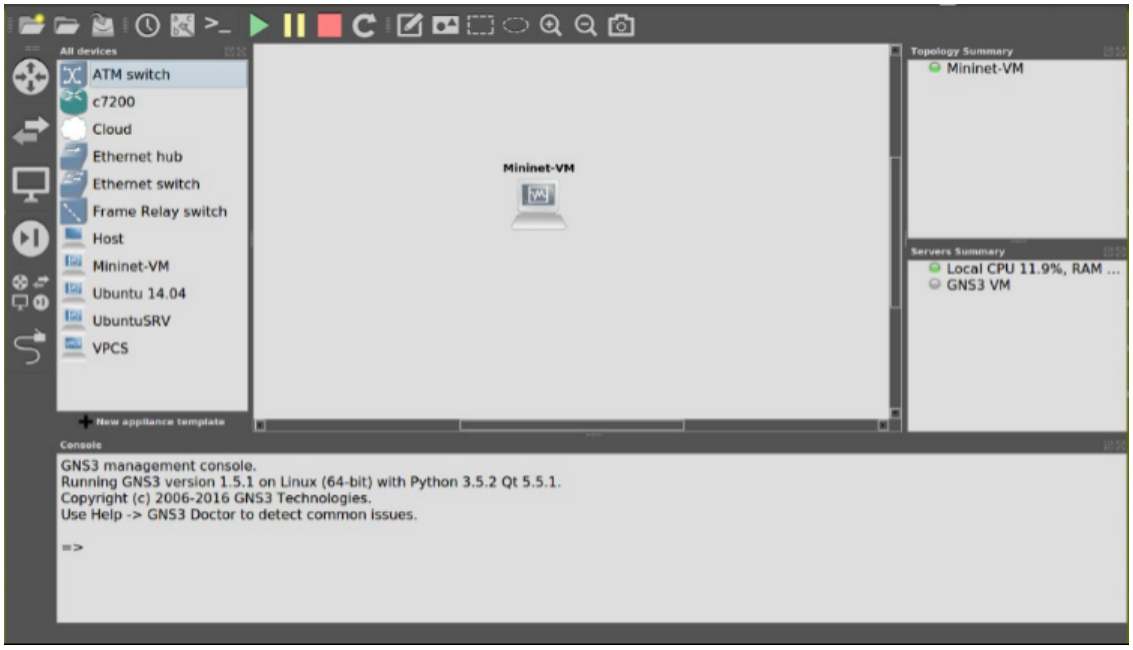


Figura 4.1: Iniciando Mininet-VM en GNS3

Por otro lado, habrá que iniciar la VM donde esté alojado OpenDayLight para tener el controlador preparado. Iniciamos ODL como se ha visto anteriormente en este trabajo. Habrá que tener en cuenta que se encuentra en la dirección 192.168.56.4:6633. Una vez iniciado, para acceder a la interfaz gráfica mediante el navedagor (DLUX) habrá que introducir la siguiente URL:

http://192.168.56.4:8181/index.html

y se mostrará la pantalla de identificación, Figura 3.16. Por defecto el usuario y la contraseña son 'admin'.

Para crear la topología que se ha propuesto, con un controlador remoto, dos switches, y dos hosts por cada switch, habrá que introducir el comando que se muestra en la Figura 4.2

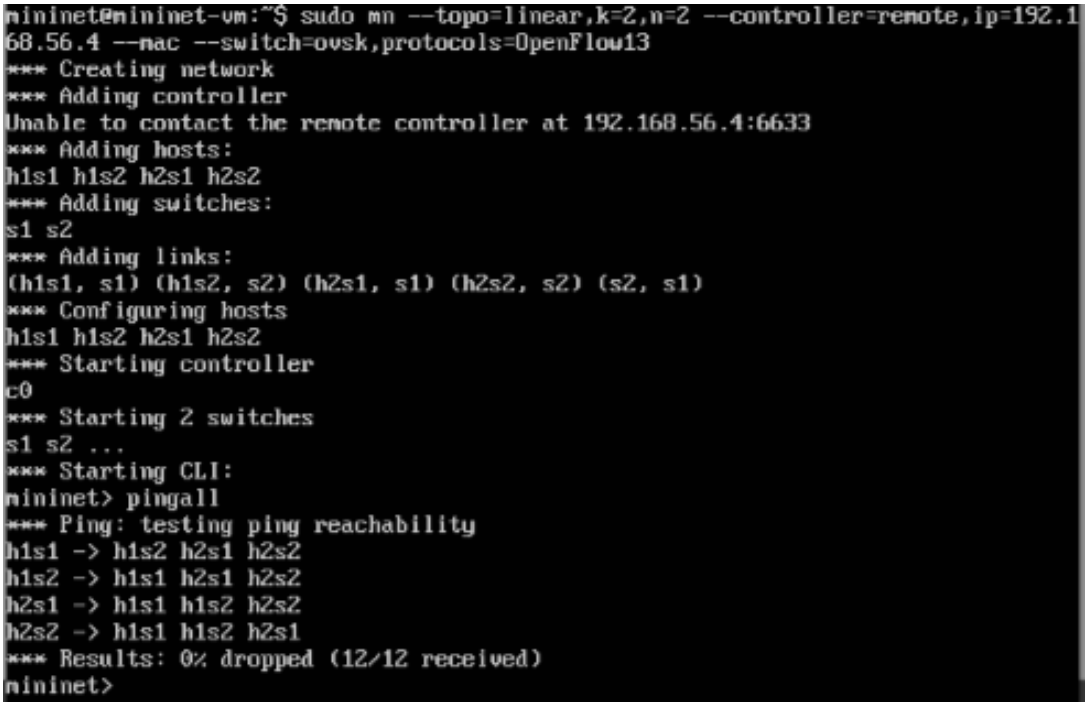


Figura 4.2: Ejecución del comando topo-linear en Mininet.

en el que:

−−**topo=linear,k=2,n=2:** permite crear una tpología con 2 switches (k) y 2 hosts conectados a cada switch (n)

−−**controller=remote,ip=192.168.56.4:** especifica que se va a utilizar un controlador externo a Mininet, alojado en esa dirección ip.

−−**mac:** ordena las direcciones MAC de los dispositivos, de forma que el primero en ser creado tendrá la 00:00:00:00:00:01 y el último la 00:00:00:00:00:04.

−−**switch=ovsk,protocols=OpenFlow13:** crea switches OpenvSwitch trabajando con la versión 1.3 de OpenFlow.

Ejecutando los comandos 'nodes', 'ports', 'net', 'intfs' y 'dump' podemos obtener datos sobre la red que nos permiten ver la estructura de la topología.

Nodes: muestra los componentes que se han creado en la emulación de la topología, como se muestra en la Figura 4.3.

Figura 4.3: Ejecucion del comando 'Nodes'.

Ports: especifica, como puede verse en la Figura 4.4, los puertos utilizados en cada interfaz para conectar los componentes.

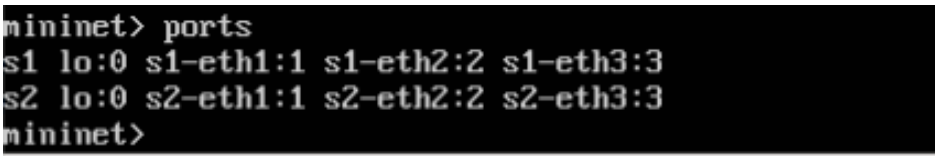


Figura 4.4: Ejecucion del comando 'Ports'.

Net: muestra los enlaces de la topología y las interfaces por las cuáles se han establecido. Figura 4.5

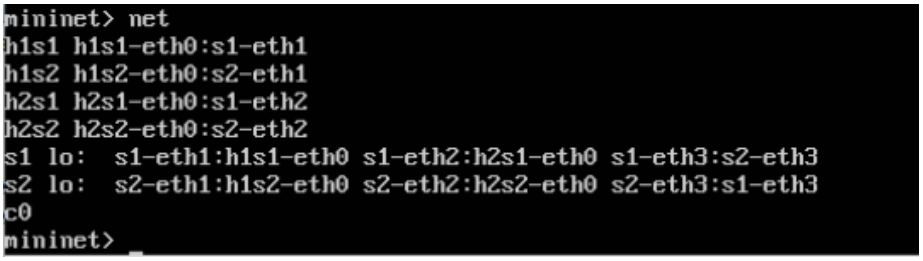


Figura 4.5: Ejecucion del comando 'Net'.

Intfs: Como se ve en la Figura 4.6, muestra las interfaces de las que dispone cada componente.

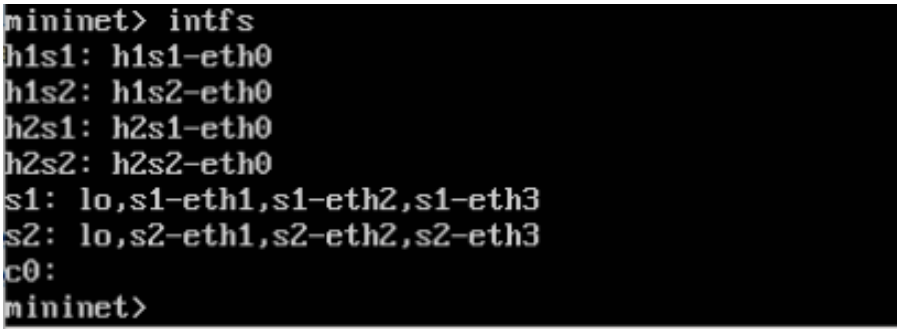


Figura 4.6: Ejecución del comando 'Intfs'.

Dump: muestra información más detallada de la red, como el nombre de cada elemento, las interfaces, las direcciones ip,... y en el caso de los siwtches, la versión de openflow que utilizan. Además de la dirección ip y el puerto en el que se encuentra el controlador, como puede apreciarse en la Figura 4.7

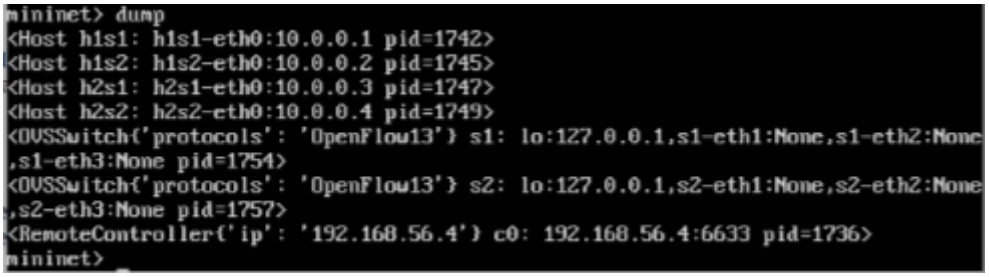


Figura 4.7: Ejecucion del comando 'Dump'.

Una vez creada la red, será necesario realizar una prueba de conexión (Figura 4.8) para comprobar que todos los hosts tienen conectividad entre sí, y para que el controlador tenga constancia de todos los nuevos elementos.

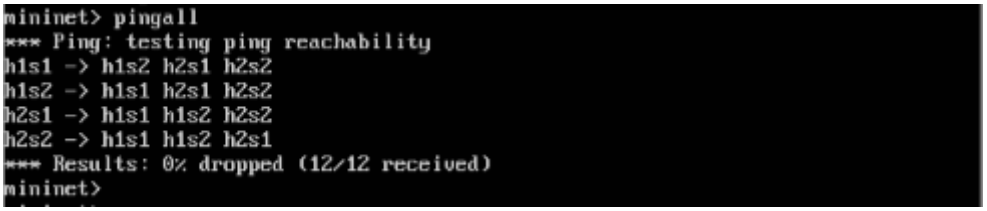


Figura 4.8: Prueba de conectividad para garantizar el correcto funcionamiento de la red.

Si todo es correcto y no hay pérdida de paquetes, se puede comprobar en la interfaz gráfica del controlador cómo efectivamente se ha creado una red (Figura 4.9) con la topología descrita:

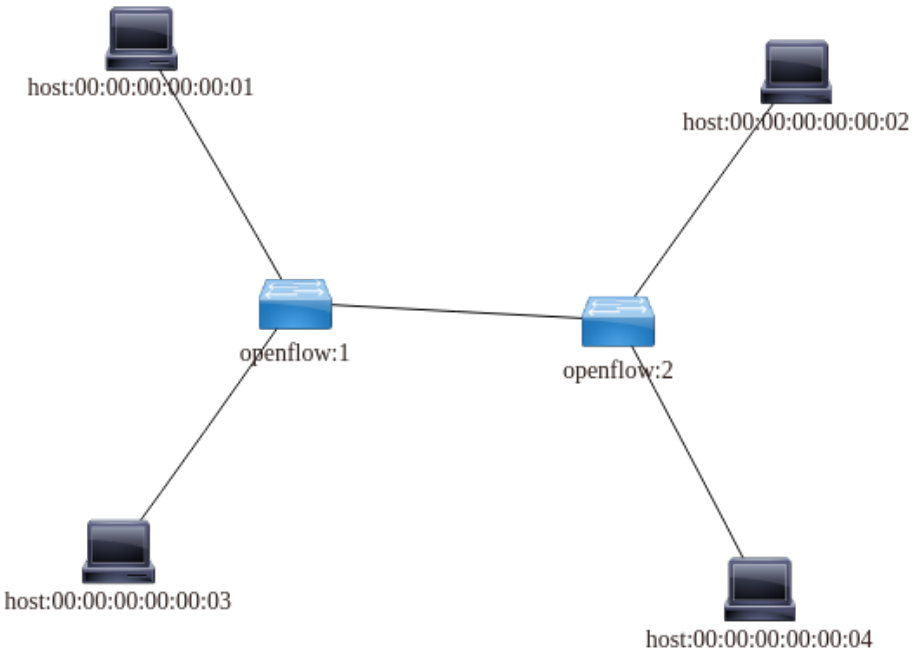


Figura 4.9: Representación de la topología creada en ODL.

4.1.2. Redes complejas utilizando Miniedit

Vista la utilidad que tiene la ventana de comandos del emulador, es momento de comprobar la potencia de la API Miniedit. Esta interfaz gráfica para generar redes facilita su creación y permite construir topologías de una dificultad mayor y más personalizadas que con la introducción de comandos por pantalla.

Partiendo de Mininet recién lanzado, como en el punto anterior, habrá que abrir un terminal para establecer una sesión SSH con una de las interfaces de la VM para poder lanzar Miniedit. En este caso la sesión SSH se establecerá con la interfaz eth1, a la dirección 192.168.57.3

```
rubenih@rubenih:~$ ssh -Y mininet@192.168.57.3
mininet@192.168.57.3's password:
Welcome to Ubuntu 14.04 LTS
(GNU/Linux 3.13.0-24-generic x86_64)
* Documentation:  https://help.ubuntu.com/
New release '16.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.
Last login: Sun Sep 11 10:15:40 2016
from 192.168.57.1
```

Una vez en mininet, habrá que ejecutar la API Miniedit:

```
mininet@mininet-vm:~$
        sudo python ./mininet/examples/miniedit.py
MiniEdit running against Mininet 2.2.1
topo=none
```

Se propone una topología compleja como la de la Figura 4.10:

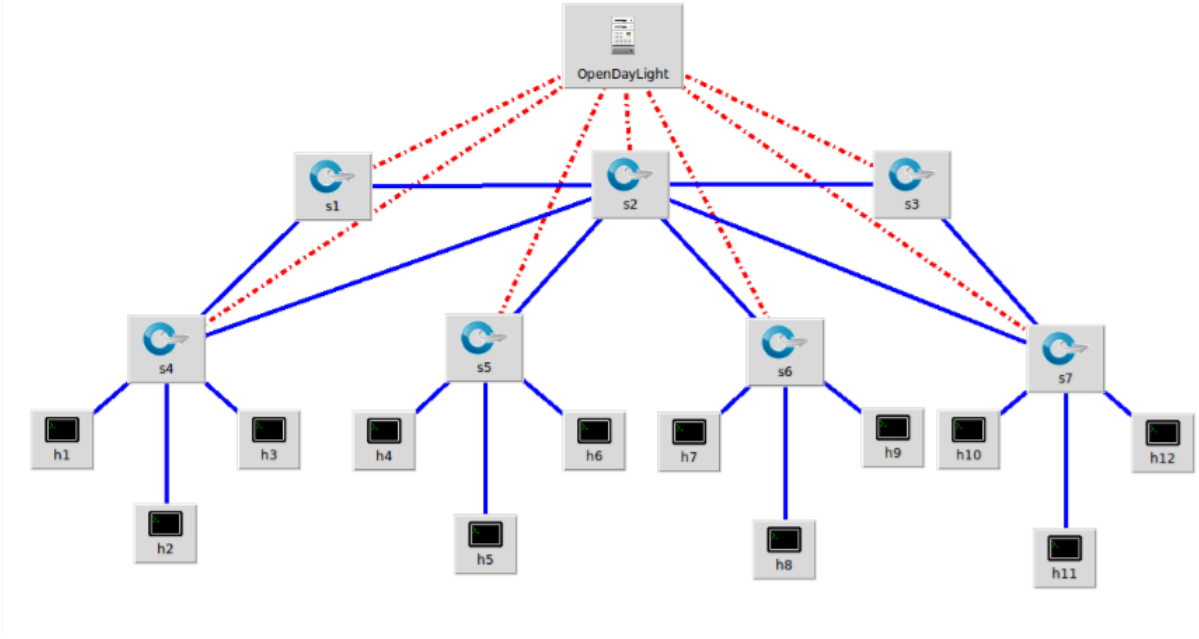


Figura 4.10: Topología propuesta para su implementación con Miniedit.

Antes de comenzar a montar la red, se pueden modificar, de forma opcional, algunos parámetros generales desde la pestaña 'Edit' de Miniedit, como muestra la Figura 4.11.

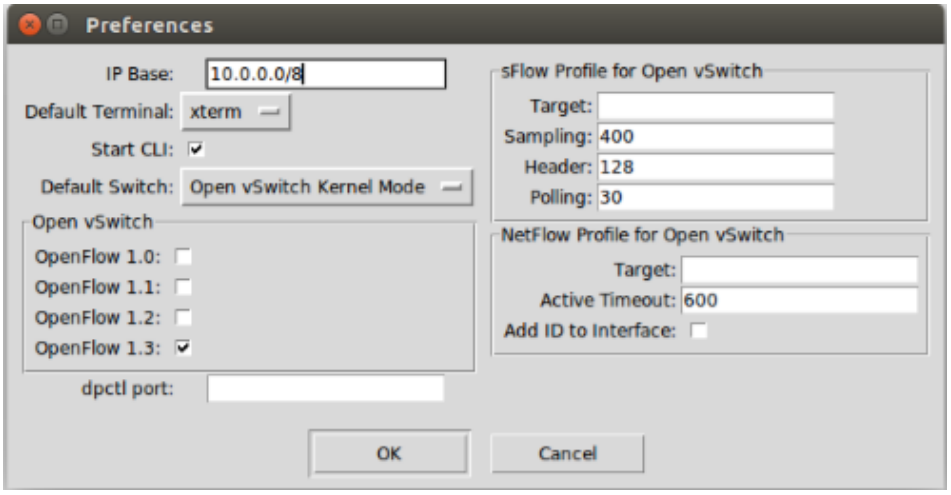


Figura 4.11: Modificación de las preferencias de Miniedit.

En este caso se ha modificado el apartado 'Start CLI' para poder sacar una línea de comandos para cada host si fuese necesario y la versión de OpenFlow, que por defecto viene en la 1.0, a la 1.3. Opciones como el rango de direcciones Ip o el tipo de switch a utilizar se han dejado por defecto.

A la hora de crear el controlador, habrá que acudir a las propiedades para modificar algunos parámetros, como en la Figura 4.12

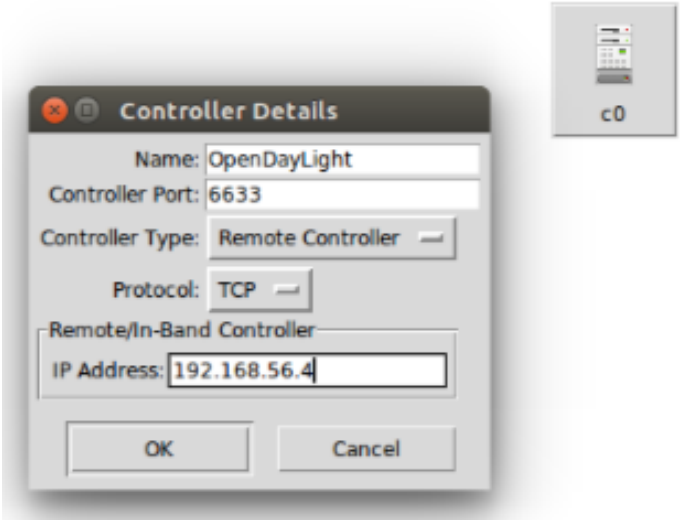


Figura 4.12: propiedades controlador

Donde se especifica que se trata de un controlador remoto, la ip donde se aloja y el puerto, que en este caso se trata del dado por defecto: 6633. El nombre se modificará de forma opcional.

A medida que se añaden elementos al área de trabajo van apareciendo líneas en la terminal donde se ejecutó la aplicación, que muestran la configuración de los componentes.

```
New controller details for OpenDayLight =
{'remotePort': 6633,
'controllerProtocol': 'tcp',
'hostname': 'OpenDayLight',
'remoteIP': '192.168.56.4',
'controllerType': 'remote'}
```

La topología propuesta tendrá varios switches OpenvSwitch conectados de forma mallada, y varios hosts por cada switch, como se muestra a continuación en la Figura 4.10.

Estas son las líneas que van apareciendo en la consola a medida que se van añadiendo componentes al diseño:

```
Open vSwitch version is 2.0.2
New Prefs = {
'ipBase': '10.0.0.0/8',
'sflow': {'sflowPolling': '30',
'sflowSampling': '400',
'sflowHeader': '128',
'sflowTarget': ''},
'terminalType': 'xterm',
'startCLI': '1', 'switchType': 'ovs',
'netflow': {'nflowAddId': '0',
'nflowTarget': '',
'nflowTimeout': '600'}, 'dpctl': '',
'openFlowVersions': {'ovsOf11': '0',
'ovsOf10': '0', 'ovsOf13': '1' }
New controller details for OpenDayLight = {
'remotePort': 6633,
'controllerProtocol': 'tcp',
'hostname': 'OpenDayLight',
'remoteIP': '192.168.56.4',
'controllerType': 'remote'}
```

Una vez conectado todo en el área de trabajo de miniedit, y con el controlador OpenDayLight lanzado y funcionando, es momento de arrancar la red apretandoel botón 'Run'. Aparecerá en la consola la configuración de la red y estará lista, si no hay errores, para trabajar con ella:

```
Build network based on our topology.
Getting Hosts and Switches.
<class 'mininet.node.Host'>
<class 'mininet.node.Host'>
<class 'mininet.node.Host'>
Getting controller selection:remote
<class 'mininet.node.Host'>
<class 'mininet.node.Host'>
<class 'mininet.node.Host'>
<class 'mininet.node.Host'>
<class 'mininet.node.Host'>
<class 'mininet.node.Host'>
<class 'mininet.node.Host'>
<class 'mininet.node.Host'>
<class 'mininet.node.Host'>
Getting Links.
*** Configuring hosts
h5 h1 h9 h10 h11 h12 h4 h2 h7 h3 h8 h6
**** Starting 1 controllers
OpenDayLight
```

```
**** Starting 7 switches
s2 s1 s4 s5 s6 s7 s3
No NetFlow targets specified.
No sFlow targets specified.


NOTE: PLEASE REMEMBER TO EXIT THE CLI BEFORE YOU
PRESS THE STOP BUTTON. Not exiting will prevent
MiniEdit from quitting and will prevent you from
starting the network again during this sessoin.

*** Starting CLI:
mininet>
```

Si se quiere comprobar por consola que todo ha sido creado como en el diseño especificado en miniedit, se puede utilizar el comando 'dump' que mostrará información de todos los componentes arrancados en ese momento:

```
mininet> dump
<Host h5: h5-eth0:10.0.0.5 pid=2169>
<Host h1: h1-eth0:10.0.0.1 pid=2173>
<Host h9: h9-eth0:10.0.0.9 pid=2176>
<Host h10: h10-eth0:10.0.0.10 pid=2192>
<Host h11: h11-eth0:10.0.0.11 pid=2199>
<Host h12: h12-eth0:10.0.0.12 pid=2206>
<Host h4: h4-eth0:10.0.0.4 pid=2213>
<Host h2: h2-eth0:10.0.0.2 pid=2216>
<Host h7: h7-eth0:10.0.0.7 pid=2219>
<Host h3: h3-eth0:10.0.0.3 pid=2226>
<Host h8: h8-eth0:10.0.0.8 pid=2229>
<Host h6: h6-eth0:10.0.0.6 pid=2236>
<customOvs s2: lo:127.0.0.1,
s2-eth1:None,s2-eth2:None,
s2-eth3:None,s2-eth4:None,
s2-eth5:None,s2-eth6:None pid=2161>
<customOvs s1: lo:127.0.0.1,
s1-eth1:None,s1-eth2:None pid=2179>
<customOvs s4: lo:127.0.0.1,
s4-eth1:None,s4-eth2:None,
s4-eth3:None,s4-eth4:None,
s4-eth5:None pid=2195>
<customOvs s5: lo:127.0.0.1,
s5-eth1:None,s5-eth2:None,
s5-eth3:None,s5-eth4:None pid=2202>
<customOvs s6: lo:127.0.0.1,
s6-eth1:None,s6-eth2:None,
s6-eth3:None,s6-eth4:None pid=2209>
```

```
<customOvs s7: lo:127.0.0.1,
s7-eth1:None,s7-eth2:None,
s7-eth3:None,s7-eth4:None,
s7-eth5:None pid=2222>
<customOvs s3: lo:127.0.0.1,
s3-eth1:None,s3-eth2:None pid=2232>
<RemoteController OpenDayLight:
192.168.56.4:6633 pid=2187>
```

Para que el controlador sea consciente de todos los elementos de la red hará falta una prueba de conectividad entre todos los nodos creados. Se utilizará el comando 'pingall':

```
mininet> pingall
*** Ping: testing ping reachability
h5 -> h1 h9 h10 h11 h12 h4 h2 h7 h3 h8 h6
h1 -> h5 h9 h10 h11 h12 h4 h2 h7 h3 h8 h6
h9 -> h5 h1 h10 h11 h12 h4 h2 h7 h3 h8 h6
h10 -> h5 h1 h9 h11 h12 h4 h2 h7 h3 h8 h6
h11 -> h5 h1 h9 h10 h12 h4 h2 h7 h3 h8 h6
h12 -> h5 h1 h9 h10 h11 h4 h2 h7 h3 h8 h6
h4 -> h5 h1 h9 h10 h11 h12 h2 h7 h3 h8 h6
h2 -> h5 h1 h9 h10 h11 h12 h4 h7 h3 h8 h6
h7 -> h5 h1 h9 h10 h11 h12 h4 h2 h3 h8 h6
h3 -> h5 h1 h9 h10 h11 h12 h4 h2 h7 h8 h6
h8 -> h5 h1 h9 h10 h11 h12 h4 h2 h7 h3 h6
h6 -> h5 h1 h9 h10 h11 h12 h4 h2 h7 h3 h8
*** Results: 0% dropped (132/132 received)
```

Si todo es correcto, desde el controlador se podrá ver la topología completa (Figura 4.13) y la información que pasa a través de los switches.

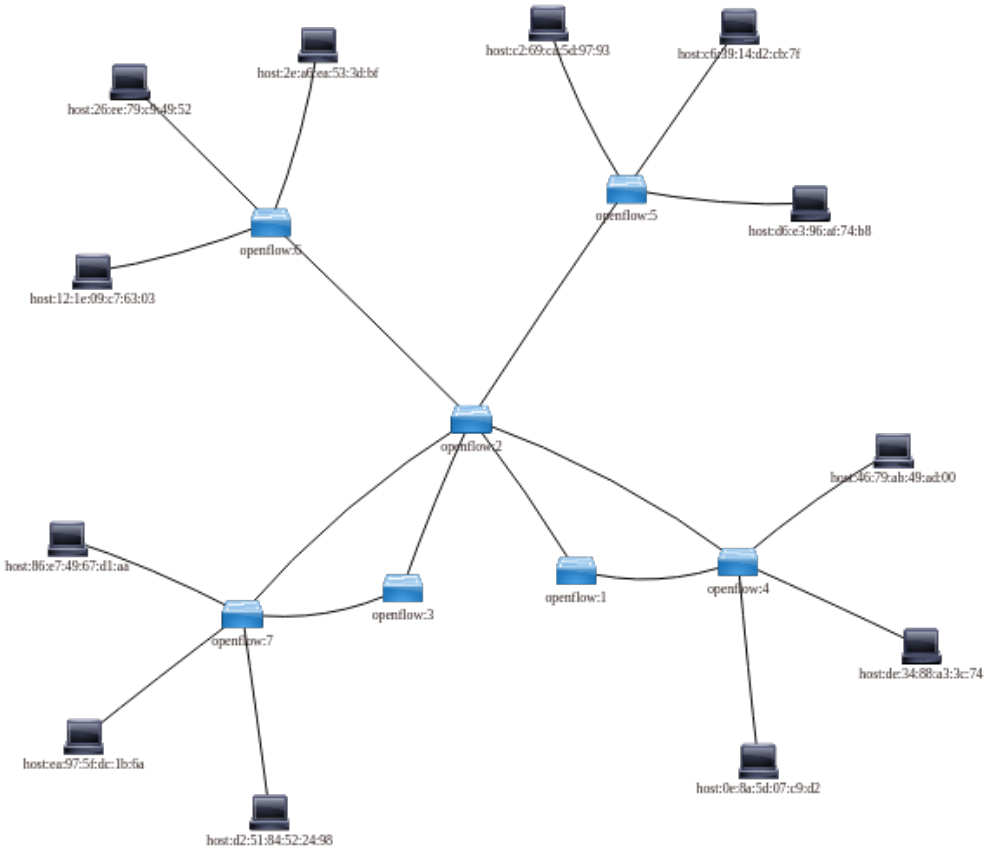


Figura 4.13: Topología creada con Miniedit vista desde el controlador ODL.

Acudiendo a la pestaña 'Nodes' se puede obtener información acerca de los siete switches generados, como se muestra en la Figura 4.14:

Node Id	Node Name	Node Connectors	Statistics
openflow:2	None	7	Flows | Node Connectors
openflow:3	None	3	Flows | Node Connectors
openflow:4	None	6	Flows | Node Connectors
openflow:5	None	5	Flows | Node Connectors
openflow:6	None	5	Flows | Node Connectors
openflow:7	None	6	Flows | Node Connectors
openflow:1	None	3	Flows | Node Connectors

Figura 4.14: Información acerca de los switches de la topología en ODL.

y pulsando sobre 'Node connectors' en cualquiera de los switches, por ejemplo el 1 (Figura 4.15), se mostrarán estadísticas del tráfico que pasa por ese nodo:

Node Connector Statistics for Node Id - openflow:1

Node Connector Id	Rx Pkts	Tx Pkts	Rx Bytes	Tx Bytes	Rx Drops	Tx Drops	Rx Errs	Tx Errs	Rx Frame Errs	Rx OverRun Errs	Rx CRC Errs	Collisions
openflow:1:LOCAL	0	0	0	0	0	0	0	0	0	0	0	0
openflow:1:2	5	8	378	633	0	0	0	0	0	0	0	0
openflow:1:1	8	8	504	633	0	0	0	0	0	0	0	0

Figura 4.15: Estadísticas de tráfico del switch 1.

Una vez se haya terminado de trabajar con esta topología. Habrá que apretar el botón 'Stop' para detenerla:

```
mininet> *** Stopping 1 controllers
OpenDayLight
*** Stopping 20 links
....................
*** Stopping 7 switches
s2 s1 s4 s5 s6 s7 s3
*** Stopping 12 hosts
h5 h1 h9 h10 h11 h12 h4 h2 h7 h3 h8 h6
*** Done
```

Adicionalmente, se puede guardar la topología para trabajar con ella desde Miniedit en otra ocasión. Habrá que acudir a File>Save y, para seguir un orden dentro de Mininet, se guardará en /home/mininet/mininet/custom con la extensión .mn, como se muestra en la Figura 4.16.

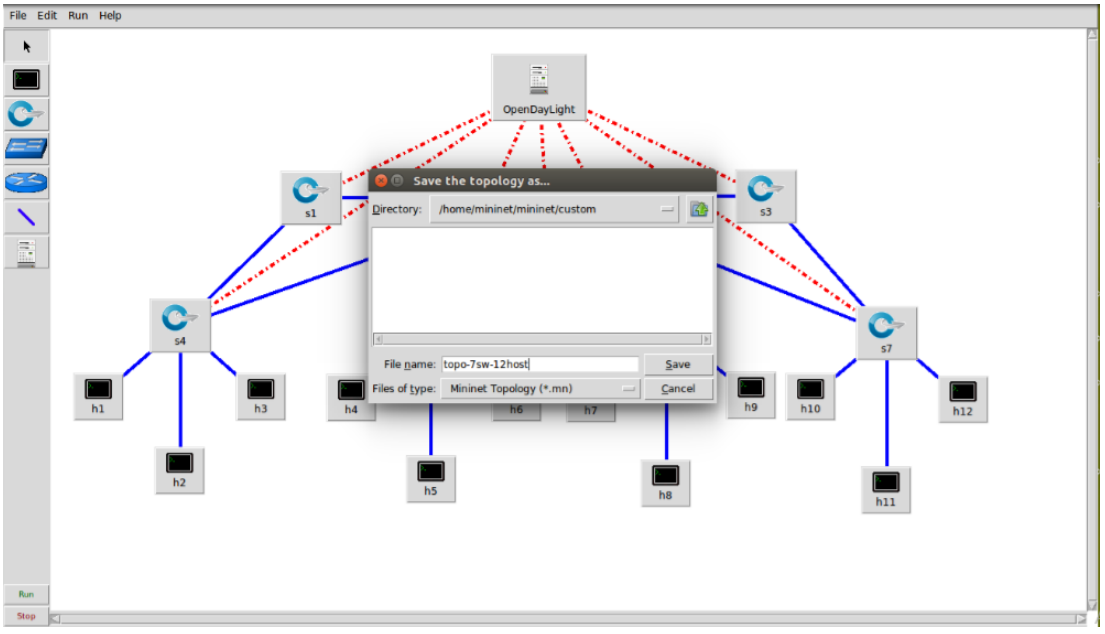


Figura 4.16: Guardando la topología con Miniedit.

Para finalizar, habrá que acudir a Mininet y, fuera de su CLI, ejecutar el siguiente comando:

```
mininet@mininet-vm:~$ sudo mn -c
*** Removing excess
controllers/ofprotocols/ofdatapaths/pings/noxes
killall controller ofprotocol ofdatapath
ping nox_core lt-nox_core ovs-openflowd
ovs-controller udpbwtest mnexec ivs 2> /dev/null
killall -9 controller ofprotocol
ofdatapath ping nox_core lt-nox_core
ovs-openflowd ovs-controller udpbwtest
mnexec ivs 2> /dev/null
pkill -9 -f "sudo mnexec"
*** Removing junk from /tmp
rm -f /tmp/vconn* /tmp/vlogs*
/tmp/*.out /tmp/*.log
*** Removing old X11 tunnels
*** Removing excess kernel datapaths
ps ax | egrep -o 'dp[0-9]+' | sed 's/dp/nl:/'
***  Removing OVS datapaths
ovs-vsctl --timeout=1 list-br
ovs-vsctl --timeout=1 list-br
*** Removing all links of the pattern foo-ethX
ip link show | egrep -o
'([-_.[:alnum:]]+-eth[[:digit:]]+)'
ip link show
*** Killing stale mininet node processes
pkill -9 -f mininet:
*** Shutting down stale tunnels
pkill -9 -f Tunnel=Ethernet
pkill -9 -f .ssh/mn
rm -f ~/.ssh/mn/*
*** Cleanup complete.
```

para eliminar cualquier rastro que hubiera podido quedar de la topología anterior y evitar conflictos en las siguientes emulaciones.

4.1.3. Ejecución de scripts en Python

Por último, se va a explicar con un ejemplo práctico cómo se crea la topología del ejemplo anterior y se ejecuta para simularla en Mininet mediante un script programado en lenguaje Python. No se profundizará en el significado de cada bloque de código, simplemente se mostrará cómo se añade cada componente y posteriormente cómo se ejecuta para poder trabajar con esa topología desde Mininet.

Se ha llamado al script 'topo-7sw-12host.py' y contiene un código que se podrá

consultar en el Anexo B.

Una vez creado, se guardará en la ruta:

```
/home/mininet/mininet/custom
```

Para ejecutar el código será necesario situarse en la carpeta contenedora

```
mininet@mininet-vm:~$ ls

index.html              loxigen   oflops   openflow
install-mininet-vm.sh   mininet   oftest   pox

mininet@mininet-vm:~$ cd mininet
mininet@mininet-vm:~$/mininet ls

bin                 custom   doc      LICENSE
mininet.egg-info    mnexec.1          setup.py
build               debian            examples
Makefile            mn.1              mnexec.c
util                CONTRIBUTORS      dist
INSTALL             mininet           mnexec
README.md

mininet@mininet-vm:~$/mininet cd custom
mininet@mininet-vm:~$/mininet/custom ls

README  topo-2sw-2host.py  topo-7sw-12host.py

mininet@mininet-vm:~$/mininet/custom
```

Y una vez dentro ejecutar:

```
mininet@mininet-vm:~$/mininet/custom
        sudo python topo-7sw-12host.py
*** Adding controller
*** Add switches
*** Add hosts
*** Add links
*** Starting network
*** Configuring hosts
h5 h1 h9 h10 h11 h12 h4 h2 h7 h3 h8 h6
*** Starting controllers
*** Starting switches
*** Post configure switches and hosts
*** Starting CLI:
mininet>
```

para lanzar la topología directamente.

Si se quiere finalizar, habrá que ordenar 'exit'

```
mininet> exit
*** Stopping 1 controllers
OpenDayLight
*** Stopping 20 links
....................
*** Stopping 7 switches
s2 s1 s4 s5 s6 s7 s3
*** Stopping 12 hosts
h5 h1 h9 h10 h11 h12 h4 h2 h7 h3 h8 h6
*** Done
mininet@mininet-vm:~$/mininet/custom
```

y volver a ejecutar el comando: sudo mn -c para limpiar los restos de la topología.

4.2. Análisis de tráfico en la SDN

De la forma en la que se ha utilizado el controlador externo ODL en este trabajo, se ha obtenido la utilidad de ver la red de forma gráfica gracias a su interfaz de usuario y los flujos de tráfico que atraviesan cada switch de la red, pero para profundizar un poco más en en el tema del análisis del tráfico que circula por la SDN, se ha optado por utilizar dos analizadores de tráfico distintos.

'tcpdump' es una herramienta para línea de comandos que ofrece la utilidad de capturar y mostrar en tiempo real los paquetes transmitidos y recibidos por la red, pudiendo aplicarse filtros para que la salida se vea más limpia.

Por otro lado se utilizará 'wireshark' , un analizador de protocolos que dará una utilidad similar a la de 'tcpdump', con el extra de añadir una interfaz gráfica y gran variedad de opciones de organización y filtrado de información. Se aplicarán los analizadores sobre una red pequeña para aprender a lanzarlos y verificar su correcto funcionamiento, así como el de la red. La red escogida para el análisis será la que Mininet toma por defecto al ejecutar:

```
mininet@mininet-vm:~$ sudo mn
*** Creating network
*** Adding controller
*** Adding hosts:                h1 h2
*** Adding switches:              s1
*** Adding links:                (h1, s1) (h2, s1)
*** Configuring hosts            h1 h2
*** Starting controller          c0
*** Starting 1 switches          s1
...
*** Starting CLI:
mininet>
```

Se trata de una topología con un controlador, un switch y dos hosts.

4.2.1. Tcpdump

Para comenzar a probar el analizador 'tcpdump' en esta topología, primero habrá que disponer de dos ventanas de comandos extra (Figura 4.17), una para cada host. Esto se consigue ejecutando el comando:

```
mininet> xterm h1 h2
```

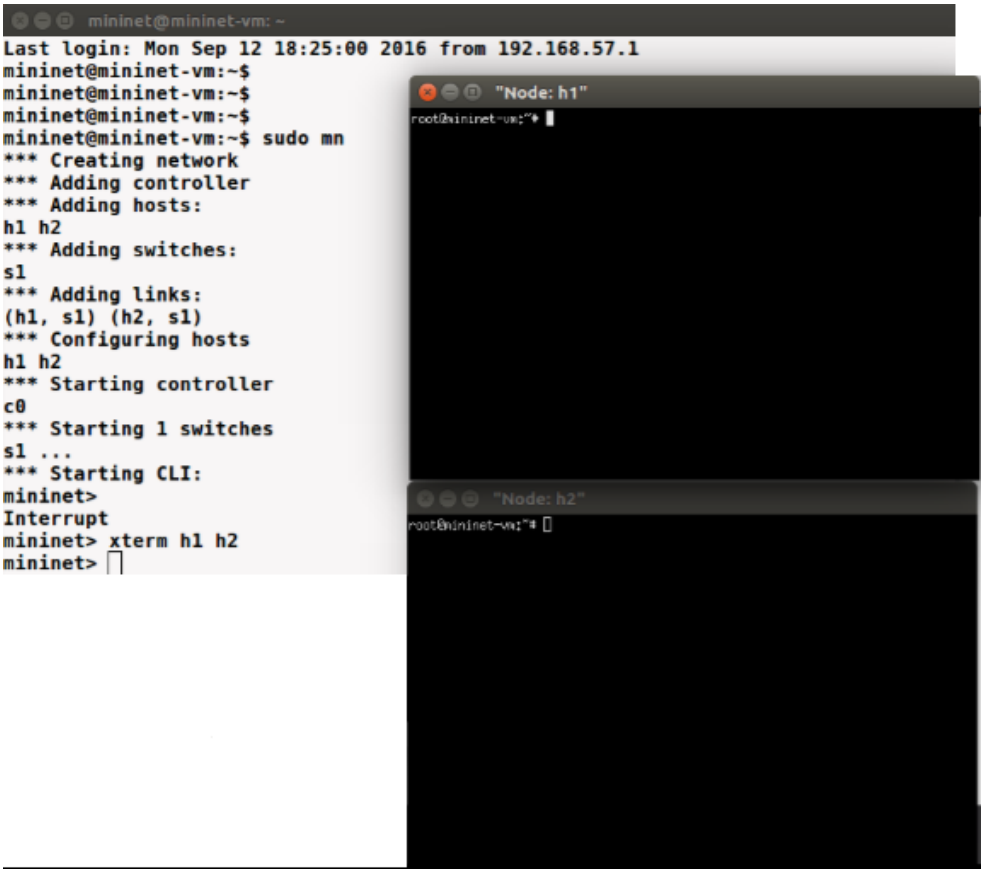


Figura 4.17: Sacar consolas de comandos para ambos hosts.

Para lanzar la herramienta basta con ejecutar el comando 'tcpdump' y se comenzará a capturar en ese terminal.

En la Figura 4.18 se ve cómo llega un ping desde h2 hasta h1, capturado con 'tcpdump'

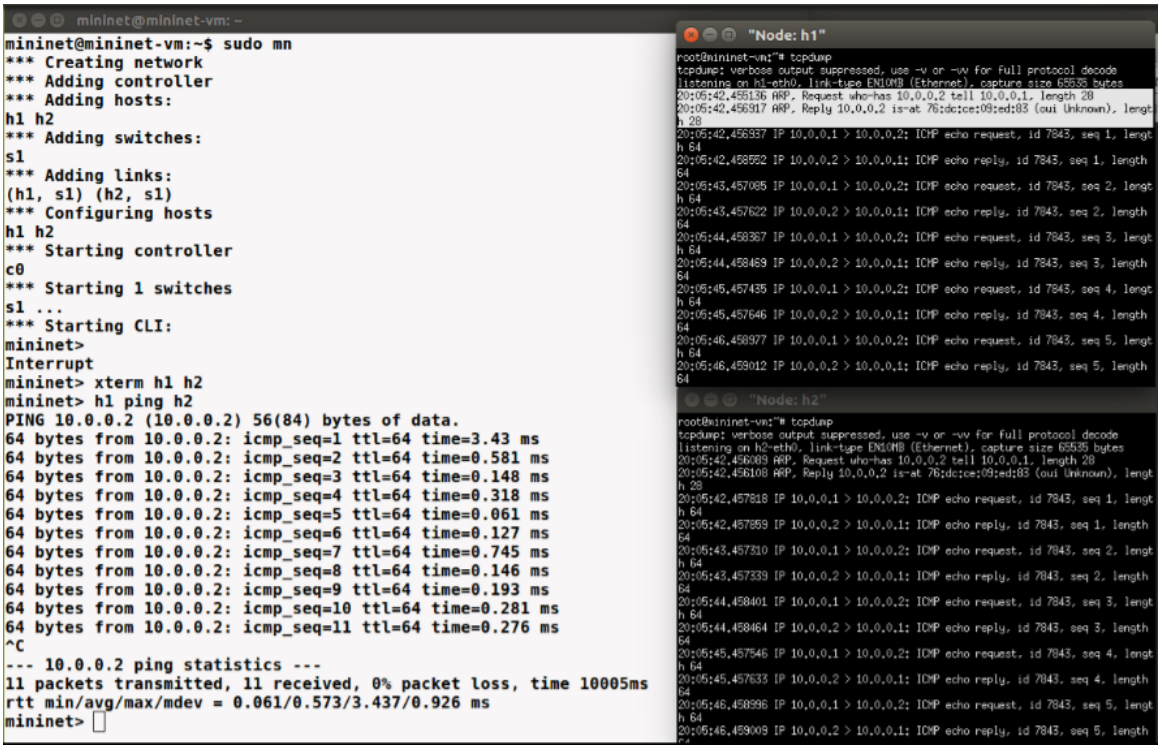


Figura 4.18: Captura de un ping desde h2 a h1 con tcpdump.

Desde la API Miniedit se pueden abrir las consolas de cada nodo simplemente haciendo click derecho sobre el componente y seleccionando la opción 'Terminal', como se puede ver en la Figura 4.19.

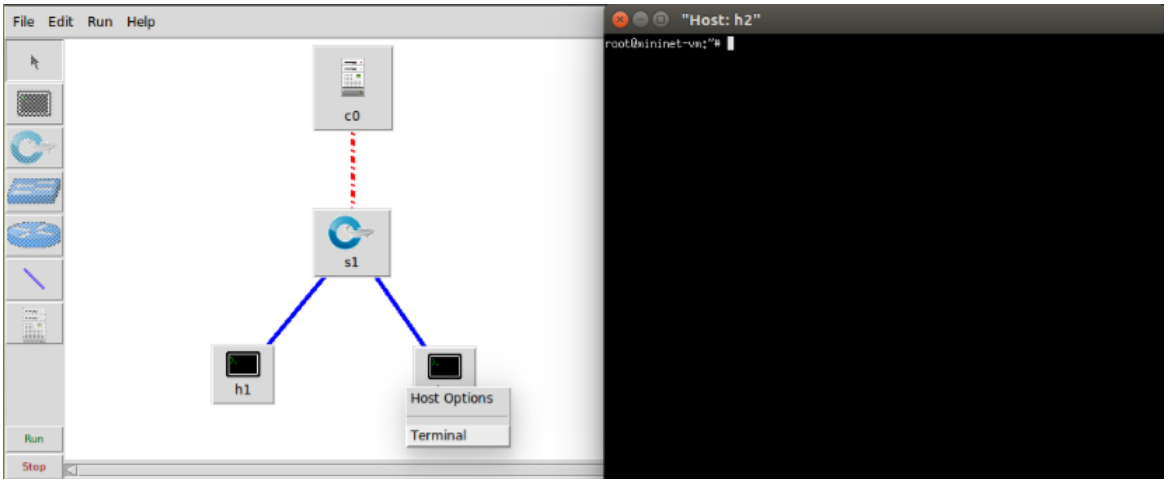


Figura 4.19: Abrir un terminal para un host desde Miniedit.

El proceso para lanzar 'tcpdump' sería análogo a lo explicado en este mismo punto.

4.2.2. WireShark

En el caso del analizador Wireshark se lanzarán también las consolas pertenecientes a cada host y se ejecutará el comando:

```
mininet> wireshark &
```

Automáticamente comenzará a ejecutarse el programa (Figura 4.20), que mostrará un mensaje de error debido la falta de un módulo, que se podrá ignorar, puesto que no interfiere con la tarea se va a realizar.

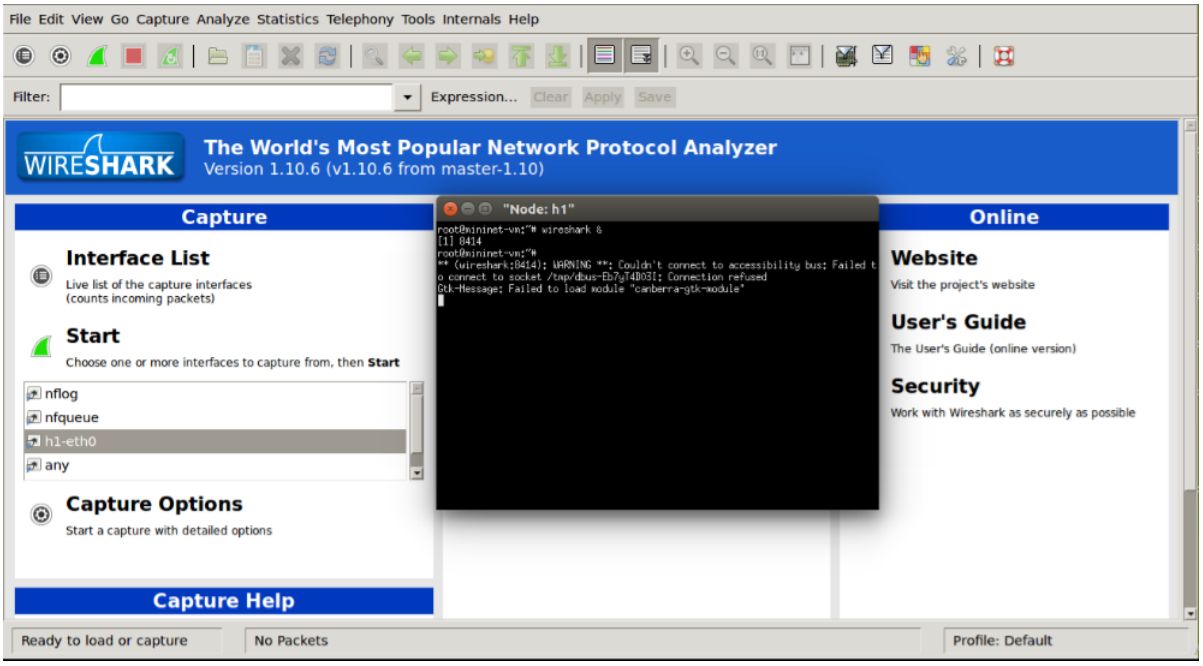


Figura 4.20: Lanzando Wireshark.

Se seleccionará de entre las interfaces que se ofrecen para capturar: '*any', y se procede a ver la captura de un ping desde h2 hacia h1.

Filtrando el tráfico, se podrá ver con claridad la información perteneciente al ping. Figuras 4.21 y 4.22.

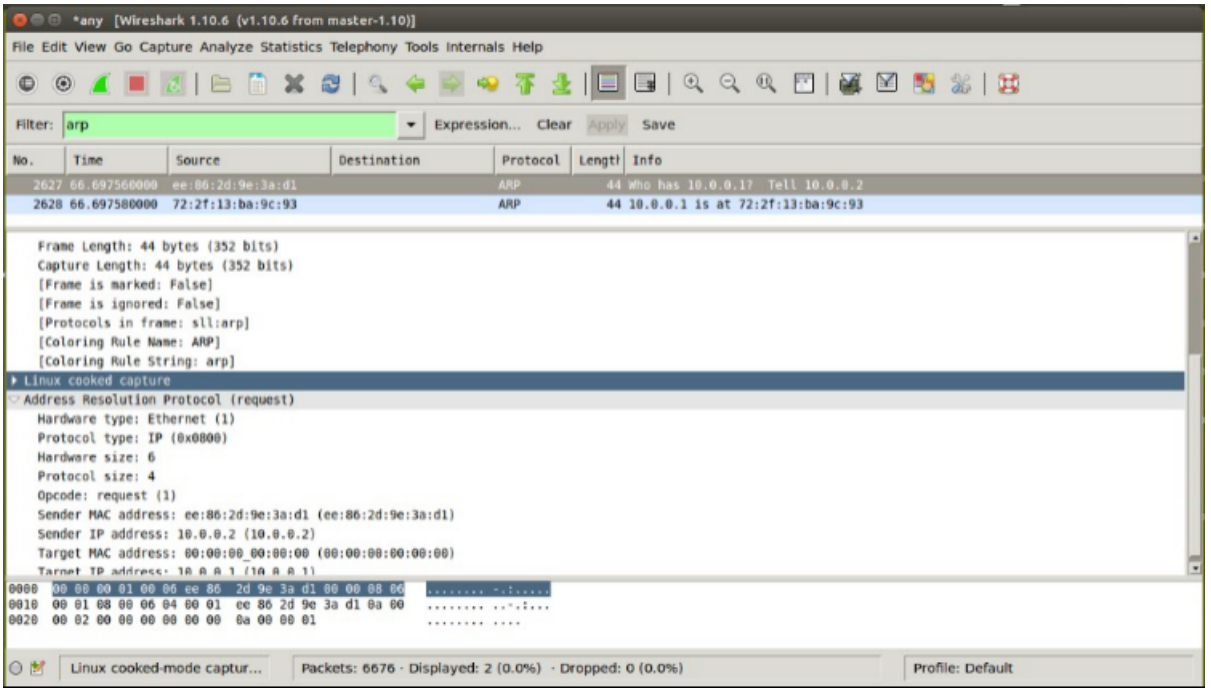


Figura 4.21: Captura en Wireshark del protocolo ARP

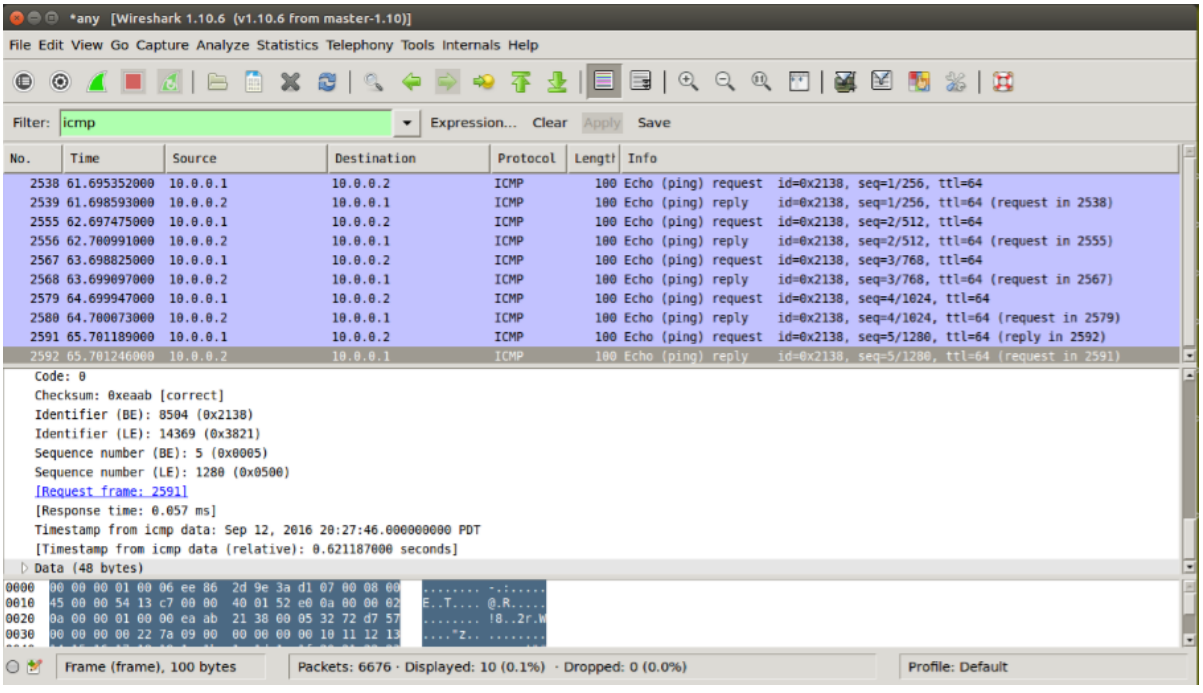


Figura 4.22: Captura en Wireshark del protocolo ICMP

Capítulo 5

Conclusiones y líneas futuras

5.1. Conclusiones

Las redes definidas por software son una tecnología aún en desarrollo que, como se ha visto a lo largo de este trabajo, pretenden ser la solución a los problemas de las actuales estructuras de red, que van apareciendo debido al incremento en la demanda de recursos por la aparición de nuevas tecnologías.

Este trabajo se ha centrado en crear un entorno virtual donde poder simular redes con este tipo de tecnología a partir de la integración de varios componentes que dota al usuario de capacidad para crear diferentes topologías de red, utilizar distintos tipos de controlador SDN: virtuales, como los que proporciona la herramienta Mininet, o reales como puede ser OpenDayLight, tener más control sobre la red que se crea gracias a Miniedit y monitorizar el tráfico que se genera en la red definida por software gracias a la integración de las herramientas de análisis de tráfico 'Wireshark' o 'tcpdump'.

En definitiva, se ha creado un entorno de trabajo que supone un buen punto de partida para todo aquél que necesite comenzar a practicar con este tipo de tencología. Sin embargo, cuenta con algunas limitaciones, como la que impone VirtualBox al sólo permitir la creación de cuatro interfaces, lo que acota la utilización de herramientas externas a Mininet si a la vez se quiere integrar en una red de mayor tamaño. En líneas futuras se trabajará en este aspecto y en la posibilidad de utilizar otro software que no imponga esta limitación.

5.2. Líneas futuras

Al tratarse un trabajo enfocado únicamente a la creación de un entorno virtual para el aprendizaje de SDN, se han dejado de hacer una serie de prácticas sobre el mismo, que si bien se escapan de los límites marcados por la tarea, habrían sido muy interesantes para comprobar alguna de las utilidades de ésta tecnología sobre un escenario simulado. Se propone la creación de la siguiente estructura de red(Figura 5.1), donde se puede ver que se conecta la red SDN, que contendrá la topología por defecto de Mininet a dos routers externos.

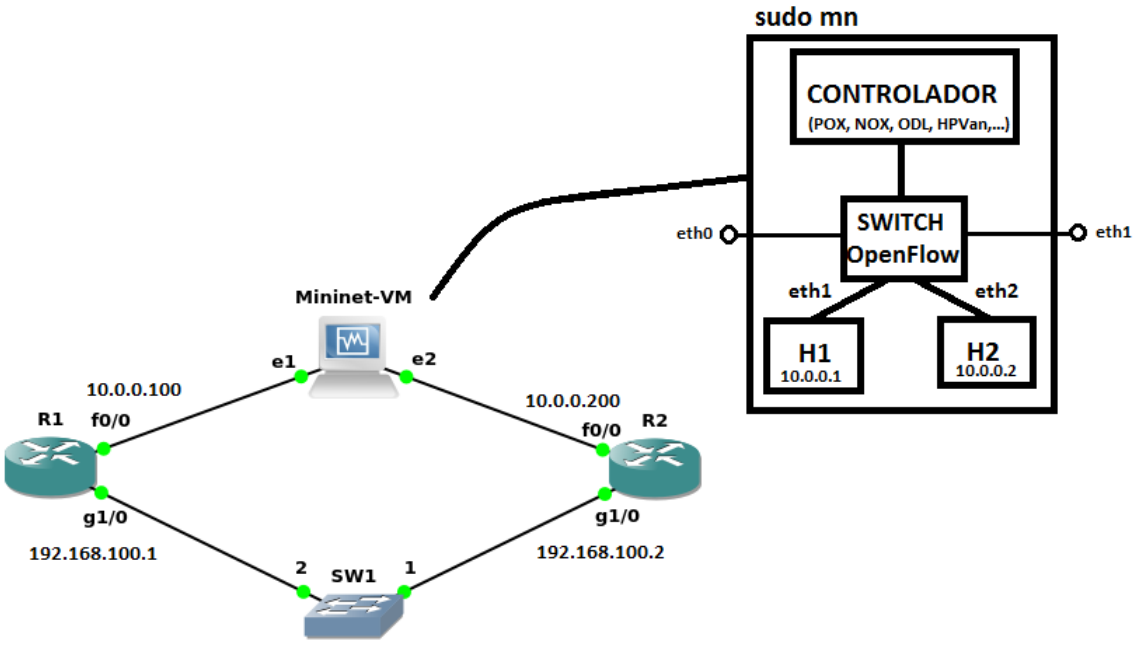


Figura 5.1: Topología de red propuesta para su simulación en líneas futuras.

La idea es conseguir, mediante la configuración de los routers mostrada en la Figura 5.2, que el tráfico generado en la red sea dirigido por el controlador de forma que los paquetes generados por el host 'H1' se encaminen por el router 'R1' y los generados desde 'H2' por el router 'R2'. Comprobando el funcionamiento mediante las herramientas de análisis de tráfico propuestas anteriormente y recibiendo los paquetes en un equipo conectado al 'SW1'.

```
R1#
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#int f0/0
R1(config-if)#ip add 10.0.0.0 255.255.255.0
Bad mask /24 for address 10.0.0.0
R1(config-if)#no shut
R1(config-if)#
*Oct  4 11:14:43.299: %LINK-3-UPDOWN: Interface FastEthernet0/0, changed state to up
R1(config-if)#
*Oct  4 11:14:43.299: %ENTITY_ALARM-6-INFO: CLEAR INFO Fa0/0 Physical Port Administrative State Down
*Oct  4 11:14:44.299: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to up
R1(config-if)#
R1#con
*Oct  4 11:14:50.643: %SYS-5-CONFIG_I: Configured from console by console
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#int g1/0
R1(config-if)#ip add 192.168.100.1 255.255.255.0
R1(config-if)#no shut
R1(config-if)#
*Oct  4 11:15:14.535: %LINK-3-UPDOWN: Interface GigabitEthernet1/0, changed state to up
R1(config-if)#
*Oct  4 11:15:14.535: %ENTITY_ALARM-6-INFO: CLEAR INFO Gi1/0 Physical Port Administrative State Down
*Oct  4 11:15:15.535: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet1/0, changed state to up
R1(config-if)#
R1#con
*Oct  4 11:15:21.795: %SYS-5-CONFIG_I: Configured from console by console
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#router rip
R1(config-router)#network 10.0.0.0
R1(config-router)#network 192.168.100.0
R1(config-router)#
R1#wr
Building configuration...

*Oct  4 11:15:42.999: %SYS-5-CONFIG_I: Configured from console by console[OK]
R1#
```

Figura 5.2: Configuración del Router R1 en la topología de red propuesta. Análogo a lo que habría que hacer con R2.

Sería un ejemplo de implementación de una red con privilegios para un tipo de usuario (H1, por ejemplo) frente a un usuario normal (H2) utilizando las redes definidas por software.

Bibliografía

[1] Open Networking Foundation, *Software-Defined Networking: The New Norm for Networks*, ONF White Paper, 2012.

[2] Ó. Roncero Hervás, *Software Defined Networking*, Universidad Politécnica de Cataluña, Máster en ingeniería telemática.

[3] Principia Technologica, *Investigación de Redes Definidas por Software*, Arquitectura SDN, 25 de marzo de 2014.

[4] Carlos Spera, *Software Defined Network: el futuro de las arquitecturas de red*, Marzo 2013.

[5] L.I. Barona López, *Propuesta de escenarios virtuales con la herramienta VNX para pruebas del protocolo OpenFlow*, 2013

[6] R. Álvarez Pinilla, *Estudio de las Redes Definidas por Software mediante el desarrollo de escenarios virtuales basados en el controlador OpenDayLight*, 2015.

[7] Ing. Jose Albert, OpenDayLight: El Futuro de las Redes Definidas por Software (SDN), DesdeLinux [Consulta el 9/10/2016] Dirección URL:http://blog.desdelinux.net/opendaylight-el-futuro-de-las-redes-definidas-por-software-sdn/Pero-que-es-OpenDayLight

[8] B. Valencia, S. Santacruz, L.Y. Becerra y J.J. Padilla, *Mininet: una herramienta versátil para emulación y prototipado de Redes Definidas por Software*, Universidad Católica de Pereira, 2015.

[9] Jackson Emilio Martínez Copete, *Estudio del funcionamiento de la herramienta Mininet*, Universidad Católica de Pereira, 2015.

[10] J. Leonardo Henao Ramírez, *Guía de implementación y uso del emulador de redes Mininet*, 2015

[11] The official GNS3 Documentation, *Linux Install Guide* [en línea] Página oficial de GNS3 [Consulta el 19/09/2016] Dirección URL: https://www.gns3.com/support/docs/linux-installation

[12] VirtualBox, Download VirtualBox [en línea] Página oficial de VirtualBox [Consulta el 19/09/2016] Dirección URL: https://www.virtualbox.org/wiki/Downloads

[13] Router vs Switch, *How to integrate GNS3 and Mininet.* [en línea] RoutervsSwitch Blogspot, Saturday, June 21, 2014 [Consulta el 19/09/2016] Dirección URL: http://router-vs-switch.blogspot.com.es/2014/06/how-to-integrate-gns3-and-mininet.html

[14] Brian Linkletter, "Open-Source Routing and Network Simulation: How to use MiniEdit, Mininet's graphical user interface" [en línea], *Brian Linkletter's Blog.* [Consulta el 19/09/2016] Dirección URL: http://www.brianlinkletter.com/how-to-use-miniedit-mininets-graphical-user-interface/

[15] OpenDayLight, *How to compile ODL on Ubuntu 14.0.4* [en línea] Wiki.OpenDayLight [Consulta el 9/10/2016] Dirección URL: https://wiki.opendaylight.org/view/Install-On-Ubuntu-14.04

Anexo A

El comando 'sudo mn', ejecutado desde una terminal Ubuntu inicia el emulador Mininet. Permite personalizar la topología y el funcionamiento de ésta mediante la introducción de diferentes opciones.

Se compone de la siguiente estructura:

sudo mn −−[OPCIÓN]=[PARÁMETRO],[ARGUMENTOS]

Las opciones que hay disponibles son:

"-h", "−−help": muestra en pantalla un listado de las posibles opciones que se pueden invocar con el comando 'sudo mn'.

"−−switch=[PARÁMETRO]" : invoca un tipo de switch compatible con Mininet de entre los siguientes:

- default: usa un switch Open vSwitch por defecto.
- ivs: IVSSwitch, es un switch OpenFlow que usa tecnología Undigo Virtual Switch y requiere instalación previa.
- ovs: Open vSwitch, usa tecnología Open vSwitch compatible con OpenFlow.
- ovsbr: OVSBridge, usa un switch Ethernet implementado a partir de Open vSwitch.
- ovsk: usa Open vSwitch en modo kernel para cada switch.
- ovsl: Open vSwitch legacy kernel-space, actualmente solo trabaja con el espacio de nombres principal Root.
- user: switch con implementación OpenFlow invocado desde el espacio de usuario, osea, externo a Mininet.
- lxbr: Linux Bridge, switch implementado en codigo abierto.

"−−host=[PARÁMETRO]": limita el ancho de banda del procesador de un host virtual:

- cfs (Completely Fair Scheduler): planificador de uso de recursos de procesamiento en Linux basado en Fair Queuing.
- rt: planificador POSIX real-time (interfaz de sistema operativo portable de tiempo real), este planificador ha sido deshabilitado por defecto en todos los kernel Linux. Se debe habilitar RT-GROUP-SCHED.

"−−controller=[PARÁMETRO]": permite la creación de un tipo de controlador compatible con Mininet de entre los siguientes:

- default: usa un controlador por defecto compatible con OpenFlow.
- none: deshabilita el uso de un controller.
- NOX: habilita un controlador tipo NOX (requiere instalación previa)
- ovsc: usa el controlados de prueba de Open vSwitch.
- remote: permite utilizar un controlador externo a Mininet y compatible con OpenFlow. Usa argumentos como
  - ip=[IP del CONTROLADOR]
  - port=[PUERTO]
- ryu: usa el controlador Ryu. (requiere instalación previa)

"−−link=[PARÁMETRO]": permite variar parámetros como ancho de banda y latencia de los enlaces:

- default: configura un enlace con ancho de banda, latencia y pérdida de paquetes por defecto.
- tc: personaliza las interfaces por medio de la utilidad Traffic Control, permitiendo especificaciones de límites de ancho de banda, latencia, pérdidas y máxima longitud de colas manejadas. Usa argumentos como:
- bw=[ANCHO DE BANDA]
- delay=[TIEMPO ms]
- loss=[PORCENTAJE]

"−−topo=[PARÁMETRO]": permite cambiar el tamaño y el tipo de topología a emular:

- linear: genera una topología de k switches en serie con n hosts conectados a cada switch. Usa argumentos como:
  - k=[NÚMERO DE SWITCHES]
  - n=[NÚMERO DE HOSTS POR SWITCH]
- minimal: genera una topología simple de dos hosts y un switch.
- single: genera una topología simple de un switch y N hosts. Usa el argumento:
  - k=[NÚMERO DE HOSTS]
- reversed: similar a una topología single, pero esta invierte el orden de los puertos en el switch. Una topología single asigna los puertos del switch en orden ascendente mientras que ésta lo hace de forma descendente. Utiliza argumentos como:
  - k=[NÚMERO DE HOSTS]

- tree: genera una topología de árbol, compuesta de N niveles (depth), N ramas (fanout) y dos hosts conectados a cada switch hoja. Usa argumentos como:
  - depth=[PROFUNDIDAD]
  - fanout=[NÚMERO DE RAMAS]
- torus: genera una topología en forma de malla MxN donde cada switch se conecta con sus vecinos más cercanos y los switches del borde se conectan con el opuesto. Mininet sólo soporta topologías 2D, este tipo de redes tiene bucles y podría no funcionar con dispositivos que no soporten el protocolo Spanning Tree, por lo tanto se recomienda utilizarla con switches lxbr, compatibles con STP. Recibe como argumentos.
  - x=[M]
  - y=[N]

"-c", "−−clean": limpia los registros de emulación y cierra el emulador Mininet.

"−−custom=[PARÁMETRO]": lee archivos de configuración de red escritos en Python, con extensión .py, para crear topologías personalizadas. Recibe como parámetro el nombre y extensión del archivo junto con su ruta relativa o absoluta.

"−−test=[PARÁMETRO]": permite realizar diferentes pruebas a la red emulada según el parámetro ingresado, estas pueden ser:

- cli: inicia la emulación y permite el uso de la línea de comandos.
- none: inicia la emulación e inmediatamente la finaliza, este parámetro puede ser usado para probar si una topología personalizada opera correctamente en el emulador.
- build: inicia la emulación hasta que la virtualización esté completamente operativa, seguidamente, se finaliza mostrando por pantalla el tiempo de duración de la ejecución.
- pingpair: inicia la emulación, realiza prueba de conectividad entre los dos primeros hosts de la emulación y la finaliza. Se usa generalmente para pruebas.
- pingall: inicia la emulación, realiza pruebas de conectividad entre todos los hosts por medio de ping y finalmente se finaliza la emulación.
- iperf: es una herramienta que mide el máximo ancho de banda entre dos hosts basado en el protocolo TCP, devolviendo información como latencia, pérdida de datagramas y ancho de banda entre dos hosts.
- iperfudp: similar al anterior, pero mide el máximo ancho de banda entre dos host basado en el protocolo UDP, retornando información como latencia, pérdida de datagramas y ancho de banda. Al terminar finaliza la emulación.

- all: inicia la simulación, realiza una prueba de conectividad entre todos los hosts, similar al funcionamiento de pingall, realiza una medición del ancho de banda entre dos hosts como lo hace iperf, y finaliza la emulación.

"-x", "−−xterms": inicia la emulación y abre una terminal independiente para cada dispotivo emulado en Mininet.

"-i [PARÁMETRO]", "−−ipbase=[PARÁMETRO]": defina el espacio de direcciones que usará le red a emular, por defecto, Mininet asigna la red 10.0.0.0/8

"−−mac": por defecto los valores de las direcciones MAC de los hosts emulados son aleatorios, este parámetro permite asignar direcciones MAC de forma ordenada.

"−−arp": Este parámetro inicializa las tablas ARP de los hosts emulados.

"-v [PARÁMETRO]","−−verbosity=[PARÁMETRO]": esta opción muestra información interna del emulador con el objetivo de depurar la operación de la virtualización, la información está clasificada en los niveles critical, error, warning, info, debug y output ordenados de mayor a menos relevancia, además, los parámetros que recibe esta opción son:

- critical, error, warning: estos parámetros retornan el mismo resultado:

```
***WARNING: selected verbosity level
(C,E,W) will hide CLI output!
Please restart Mininet with -v [debug,info,
output].
```

- info: este parámetro es ejecutado por defecto, permite visualizar en consola lo que el emulador está realizando durante el inicio y finalización de la emulación.
- output: este parámetro reduce la información visualizada en consola a sólo los datos necesarios.
- debug: este parámetro habilita una visualización muy detallada sobre el comportamiento del emulador Mininet durante toda la ejecución.

"−−innamespace": por defecto los hosts están puestos en su propio espacio de nombres, mientras que switches y controladores están en el espacio de nombres principal (root), este parámetro ubica a los switches en su propio espacio de nombres permitiendo la separación en el sistema de comunicación entre switches y controladores.

"−−listenport=[PARÁMETRO]": esta opción recibe como parámetro un número que será la base para estableces los puertos lógicos de los switches. Por defecto, el puerto lógico asignado a un switch es el 6634 e incrementa según el número de switches usados.

“−−nolistenport”: deshabilita el uso de un puerto lógico en los switches emulados inclusive el puerto lógico por defecto 6634.

“−−nat”: agrega servicio NAT entre la red emulada y el host anfitrión, este parámetro ofrece conectividad entre los dispositivos emulados en Mininet y el host anfitrión.

“−−version”: muestra por pantalla el número de versión de Mininet.

Se detallan a continuación los comandos para la CLI (línea de introducción de comandos) de Mininet.

mininet> EOF: el comando “EOF” finaliza la emulación de Mininet.

mininet> exit: el comando “exit” finaliza la emulación y cierra el programa.

mininet> quit: el comando “quit” ” finaliza la emulación actual.

mininet> help: el comando “help” muestra en pantalla documentación e información del uso de comandos.

mininet> dump: el comando “dump” muestra en pantalla información detallada de la red, datos como tipo de dispositivo, nombre, puerto usado, dirección IP e ID de proceso.

mininet> net: el comando “net” muestra en pantalla los enlaces y los respectivos puertos usados por los dispositivos emulados.

mininet> intfs: el comando “intfs” detalla las interfaces usadas por los dispositivos emulados.

mininet> nodes: el comando “nodes” muestra en pantalla los nodos emulados.

mininet> ports: el comando “ports” muestra en pantalla los puertos e interfaces de cada switch emulado.

mininet> time [comando]: el comando “time” muestra en pantalla el tiempo que usa cualquier comando Mininet para ejecutarse.

mininet> switch [switch] [start/stop]: el comando “switch” inicia o detiene el funcionamiento del switch especificado.

mininet> links: el comando “links” reporta los enlaces que presentan un estado correcto y funcional.

mininet> link [nodo1] [nodo2] [up/down]: el comando “link” habilita o deshabilita el enlace entre dos nodos.

mininet> noecho [host] [cmd args]: el comando “noecho” ejecuta acciones y comandos directamente en el dispositivo virtualizado sin realizar eco en el host anfitrión de Mininet, evita el uso de herramientas como xterm y gterm.

mininet> sh [cmd args]: el comando "sh" ejecuta un comando Shell del sistema operativo Ubuntu.

mininet> source ¡file¿: el comando "source" lee comandos Mininet desde un archivo de entrada; habrá que tener en cuenta el manejo de rutas absolutas o relativas.

mininet> pingall: el comando "pingall" realiza una prueba de conectividad entre todos los host emulados.

mininet> pingallfull: el comando "pingallfull" realiza una prueba de conectividad entre todos los host emulados, mostrando en pantalla los resultados.

mininet> pingpair: el comando "pingpair" realiza una prueba de conectividad entre los dos primeros host emulados (h1 y h2).

mininet> pingpairfull: el comando "pingpairfull" realiza una prueba de conectividad entre los dos primeros host emulados (h1 y h2) mostrando en pantalla los resultados.

mininet> iperf [host1] [host2]: el comando "iperf" es una herramienta que prueba del rendimiento de ancho de banda TCP entre dos host específicos.

mininet> iperfudp [bw] [host1] [host2]: el comando "iperfudp" es una herramienta que prueba el rendimiento de ancho de banda UDP entre dos host específicos.

mininet> px [PYTHON]: el comando "px" ejecuta declaraciones en lenguaje de programación Python, es posible usar variables y funciones de las librerías de Mininet.

mininet> py [OBJETO.FUNCION()]: el comando "py" permite evaluar y ejecutar desde la consola Mininet expresiones en lenguaje Python y basadas en librerías Mininet.

mininet> xterm [nodo1] [nodo...]: el comando "xterm" abre una nueva terminal para los nodos especificados.

mininet> x [host] [cmd args]: el comando "x" crea un túnel X11 a un host específico.

mininet> gterm [nodo1] [nodo...]: el comando "gterm" abre terminales-gnome para los host solicitados.

mininet> dpctl [COMANDO][ARGUMENTOS]: dpctl (o ovs-ofctl) es una herramienta de administración y monitoreo que se ejecuta en todos los switches OpenFlow emulados cuando es invocado desde la consola Mininet.

Anexo B

Se muestra el código en Python para la implementación de la topología propuesta en el apartado 4.1.3.

```
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( '*** Agregando controlador\n' )
OpenDayLight=net.addController(name='OpenDayLight',
controller=RemoteController,
ip='192.168.56.4',
protocol='tcp',
port=6633)

info( '*** Agregando switches\n')
s2 = net.addSwitch('s2', cls=OVSKernelSwitch)
s1 = net.addSwitch('s1', cls=OVSKernelSwitch)
s4 = net.addSwitch('s4', cls=OVSKernelSwitch)
s5 = net.addSwitch('s5', cls=OVSKernelSwitch)
s6 = net.addSwitch('s6', cls=OVSKernelSwitch)
s7 = net.addSwitch('s7', cls=OVSKernelSwitch)
s3 = net.addSwitch('s3', cls=OVSKernelSwitch)
```

```
info( '*** Agregando hosts\n')
h5 = net.addHost('h5', cls=Host, ip='10.0.0.5',
defaultRoute=None)
h1 = net.addHost('h1', cls=Host, ip='10.0.0.1',
defaultRoute=None)
h9 = net.addHost('h9', cls=Host, ip='10.0.0.9',
defaultRoute=None)
h10 = net.addHost('h10', cls=Host, ip='10.0.0.10',
defaultRoute=None)
h11 = net.addHost('h11', cls=Host, ip='10.0.0.11',
defaultRoute=None)
h12 = net.addHost('h12', cls=Host, ip='10.0.0.12',
defaultRoute=None)
h4 = net.addHost('h4', cls=Host, ip='10.0.0.4',
defaultRoute=None)
h2 = net.addHost('h2', cls=Host, ip='10.0.0.2',
defaultRoute=None)
h7 = net.addHost('h7', cls=Host, ip='10.0.0.7',
defaultRoute=None)
h3 = net.addHost('h3', cls=Host, ip='10.0.0.3',
defaultRoute=None)
h8 = net.addHost('h8', cls=Host, ip='10.0.0.8',
defaultRoute=None)
h6 = net.addHost('h6', cls=Host, ip='10.0.0.6',
defaultRoute=None)

info( '*** Agregando enlaces\n')
net.addLink(s1, s2)
net.addLink(s2, s3)
net.addLink(s1, s4)
net.addLink(s2, s5)
net.addLink(s2, s6)
net.addLink(s2, s4)
net.addLink(s2, s7)
net.addLink(s3, s7)
net.addLink(h1, s4)
net.addLink(h2, s4)
net.addLink(s4, h3)
net.addLink(h4, s5)
net.addLink(s5, h5)
net.addLink(s5, h6)
net.addLink(h7, s6)
net.addLink(s6, h8)
net.addLink(s6, h9)
net.addLink(h10, s7)
```

```
net.addLink(s7, h11)
net.addLink(s7, h12)

info( '*** Iniciando la red\n')
net.build()
info( '*** Iniciando controladores\n')
for controller in net.controllers:
controller.start()

info( '*** Iniciando switches\n')
net.get('s2').start([OpenDayLight])
net.get('s1').start([OpenDayLight])
net.get('s4').start([OpenDayLight])
net.get('s5').start([OpenDayLight])
net.get('s6').start([OpenDayLight])
net.get('s7').start([OpenDayLight])
net.get('s3').start([OpenDayLight])

CLI(net)
net.stop()

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

Lista de Acrónimos

SDN: Software Defined Network(Red definida por software).

ODL: OpenDayLight

API: Application Programming Interface (Interfaz de programación de aplicaciones).

RFC: Requests for comments (Solicitudes de comentarios).

SO: Sistema Operativo.

CLI: Command Line Interface (Interfaz de línea de comandos).

IOS: Internetwork Operating System.

VM: Virtual Machine (Máquina virtual).

NAT: Network Address Translation (Traductor de direcciones de red).

GUI: Graphical User Interface (Interfaz gráfica de usuario).

UI: User Interface(Interfaz de usuario).

URL: Uniform Resource Locator (Localizador de recursos uniforme).

SNMP: Simple Network Management Protocol (Protocolo simple de administración de red).

REST: Representational State Transfer.

SSH: Secure SHell (Intérprete de órdenes seguro).