Universidad Zaragoza
1542

Trabajo Fin de Grado

Implementación y evaluación de un sistema SD-WAN para un entorno empresarial virtualizado

Implementation and evaluation of an SD-WAN system for a virtualized business environment

Autor

Javier Ramos Fernández

Director

Julián Fernández Navajas

Escuela de Ingeniería y Arquitectura / Universidad de Zaragoza

2019

Agradecimientos:

A Julián, por su ayuda y consejo.
A todos los que me han apoyado.
A mi madre Ana, mi padre Evelio y mi hermano Martín, junto con el resto de familia y amigos, que siempre confían en mí.

Implementación y evaluación de un sistema SD-WAN para un entorno empresarial virtualizado

RESUMEN

Este Trabajo Fin de Grado se ha fundamentado en la necesidad de la empresa **EFOR Internet y Tecnología** por estudiar las posibilidades que nos brinda la tecnología **SD-WAN** *(Software Defined in Wide Area Networks).* Para ello, se ha trabajado en un entorno virtualizado, creando los escenarios de red necesarios, sobre los cuáles se han realizado pruebas para analizar el correcto funcionamiento y las posibles ventajas que nos ofrece esta tecnología.

Existen soluciones propietarias de grandes empresas, pero el objetivo de este TFG no es implementar y analizar ninguna de estas soluciones. Siendo conocedores de su existencia y características, el objetivo es analizar el uso de **SDN** *(Software Defined Network)* en entornos generales y de menor complejidad, estudiando la tecnología que hay detrás y si realmente existen ventajas y mejores rendimientos mediante su implantación.

A lo largo de la memoria se describen las herramientas utilizadas, se realiza una descripción de los escenarios creados y se presentan las diferentes pruebas a las que han sido sometidos. Se comprobará el funcionamiento dinámico de la comunicación, para gestionar y analizar diferentes tipos de tráficos que atraviesan la red.

Para facilitar el análisis, en el presente Trabajo se han descrito las configuraciones con la mayor sencillez posible para beneficio del lector. Se destacan ilustraciones de los escenarios y código de configuración de todas las máquinas. Cabe destacar que estas máquinas están equipadas con software libre, para facilitar futuras continuaciones del proyecto.

Finalmente, en esta memoria, se recogen todas las conclusiones y posibles líneas futuras de este Trabajo Fin de Grado.

ÍNDICE

ÍNDICE DE ILUSTRACIONES

1. INTRODUCCIÓN

1.1. INTRODUCCIÓN

Este Trabajo Fin de Grado está realizado en la empresa aragonesa **EFOR Internet y Tecnología.** EFOR, está presente en Madrid, Barcelona, Zaragoza y Bilbao, y con intención de seguir expandiéndose tanto nacional como internacionalmente. Son especialistas en soluciones de software, marketing online, proyectos de *IoT (Internet of Things)*, sistemas, infraestructura tecnológica (IT), gestión, formación, soluciones *cloud*, etc. A su vez, EFOR es una división de **Integra - Estrategia y Tecnología,** que es una consultora tecnológica formada por diferentes marcas de referencia en varios sectores, que crea, a través de la especialización en áreas tecnológicas y estratégicas, un entorno de valor diferencial para sus clientes.

En el seno de EFOR, nace la posibilidad de implementar y evaluar sistemas ***SD-WAN*** *(Software Defined – Wide Area Network)* para entornos empresariales. En particular, desde su área de "IoT y Comunicaciones", surge la necesidad de analizar el comportamiento de las redes *SD-WAN*, para continuar creciendo y hacer posible la incorporación de nuevos servicios y productos a su oferta.

Se puede consultar más información sobre esta empresa en las páginas web de https://www.integratecnologia.es [1] y https://www.efor.es/ [2]

Para introducirnos en la materia, definimos el termino SD-WAN cómo la aplicación de la tecnología **SDN** (*Software Defined Networking*) a las redes **WAN** *(Wide Area Network)*, que son redes de computadores que unen varias redes locales, aunque sus miembros no estén todos en una misma ubicación física.

Vemos entonces que el concepto clave de esta tecnología son "las **redes definidas por software** (*SDN*)", qué "son un conjunto de técnicas relacionadas con el área de redes computacionales, cuyo objetivo es facilitar la implementación e implantación de servicios de red de una manera determinista, dinámica y escalable, evitando al administrador de red gestionar dichos servicios a bajo nivel. Todo esto se consigue mediante la separación del plano de control (software) del plano de datos (hardware)." (IETF 7149: Software-Defined Networking: A Perspective from within a Service Provider Environment). [3]

Destacamos de la definición, que se evita al administrador de red gestionar la implementación e implantación de servicios de red a bajo nivel. Además, aparece una nueva figura llamada "controlador", común a varios equipos de la red y que mediante software realizará las funciones de control sobre el tráfico de la red. Por tanto, los conmutadores dejarán de tomar decisiones por ellos mismos y siempre se apoyarán en el controlador, preguntándole qué decisiones deben tomar sobre el tráfico que pase por ellos. De esta manera, el controlador conocerá el estado de la red y decidirá cuál será la mejor actuación en cada momento.

A la hora de montar y gestionar una red de estas características, aparecen una serie de ventajas y desventajas. Algunas de las ventajas que se destacan son:

- **Simplicidad de los equipos.** Podemos utilizar hardware más sencillo, ya que ahora la mayor parte del peso del plano de control de la red recae sobre el controlador.

- **La escalabilidad de la red.** Con la posibilidad de usar equipos más simples y gran parte del control desvinculado de estos equipos, se facilita la ampliación o reducción de la red.

- **Encaminamiento dinámico.** Al centralizar el control del tráfico en la red, podemos modificar dinámicamente el encaminamiento, para qué los usuarios puedan beneficiarse de menores retardos y pérdidas de acuerdo con las decisiones del controlador, que tiene una visión global de la red.

Por otro lado, también existen desventajas y el principal podría ser:

- **Centralización.** Aunque en principio pueda verse como una ventaja para la monitorización y toma de decisiones de toda la red en tiempo real, que todo el tráfico de la red dependa de un solo nodo puede resultar arriesgado. Dado que, si este cae, los equipos no obtendrán respuesta por parte del controlador caído de las decisiones que debe tomar, por lo que el tráfico se perdería.

  Aunque esta desventaja se percibe a priori como algo grave, existen técnicas de *backup* para ceder el control a otro equipo controlador en caso de pérdida de conexión. También se pueden definir configuraciones con las que los equipos puedan continuar actuando en función de cómo lo estaban haciendo hasta el momento, o incluso que puedan intentar interpretar las rutas que debe seguir cada paquete, como si ellos conocieran el estado de la red. Todo dependerá del riesgo que se esté dispuesto a asumir si esta situación llega a ocurrir.

Existen protocolos normalizados para establecer la comunicación entre los equipos y el controlador, por ejemplo, OpenFlow es uno de ellos. Aunque con él se podrán realizar configuraciones básicas, existirán redes con características particulares en las que este protocolo general no consiga dar respuesta. También se sabe de la existencia de soluciones propietarias en el mercado, las cuales EFOR conoce y podría comenzar a utilizar y distribuir. [4]

Por último, conocemos varias herramientas de virtualización, con las que crear escenarios de red, sin riesgo de cometer errores críticos. Una de las más utilizadas es GNS3. Esta herramienta nos permite crear un escenario de red virtualizado que podremos configurar de principio a fin y con el que poder realizar pruebas controladas. Evitando así la intervención de agentes externos, además de los problemas que pueden suponer realizar una implementación en un escenario real, cómo por ejemplo cortes en la red, perdidas de datos, etc.

1.2. MOTIVACIÓN

El SD-WAN es una tecnología de la que se lleva hablando durante algunos años. Aunque teóricamente son numerosas las ventajas, a muchas empresas les cuesta dar el salto y realizar cambios profundos en sus comunicaciones. Por este motivo, EFOR quiere especializarse en esta tecnología, para poder seguir creciendo y abrir aún más su abanico de productos, pudiendo ofrecer tecnología *SD-WAN* a sus clientes.

Pero antes de comenzar a explotarla, tiene la necesidad de probar y experimentar, con el fin de verificar su efectividad. Para ello les gustaría hacerlo en un entorno controlado, en el que no exista peligro alguno de perdida de información, ni caída de servicios, etc. En definitiva, que no altere ninguna comunicación real.

Otro motivo por el que en EFOR están interesados en esta tecnología, es el ahorro de costes. Ya que esta tecnología puede aplicarse en entornos y dispositivos con recursos limitados, para una primera etapa se está pensando en implementarla en redes formadas por equipos con un poder computacional reducido. Estos equipos pueden limitarnos a la hora de realizar determinadas configuraciones, pero a su vez, nos brinda la posibilidad de posteriormente escalar el proyecto en dimensiones y complejidad.

1.3. OBJETIVOS

Existen numerosas posibles soluciones para implementar un escenario con todas las características que ofrece el *SD-WAN*, por ejemplo, desarrollarlo en una red ya existente, para analizarla y realizar pruebas. Esta se descarta, porque generaría multitud de problemas, como cortes en la red, configuraciones erróneas, etc. Otra opción podría ser la creación de una red real únicamente para este propósito, pero tendría un coste económico demasiado elevado. Por estos motivos, se decide crear un entorno virtualizado.

El objetivo de este Trabajo Fin de Grado es el análisis de los sistemas definidos por software y su aplicación en entornos de redes de área amplia (SD-WAN). Tras la creación de un entorno virtualizado completo que facilite la realización de pruebas controladas para la posterior captura y procesamiento de datos, estos se analizarán para evaluar el correcto control aplicado a cada tipo de tráfico entre los diferentes enlaces y en diferentes situaciones.

La red estará formada por *switch Ethernet* con la capacidad de ser programados y por generadores de tráfico. Queremos ser capaces de proporcionar comunicación a todo nuestro escenario virtualizado y, además, controlar el tráfico de éste mediante el uso de órdenes concretas para cada flujo de datos. Primero, comprobando el correcto funcionamiento de las órdenes directamente en los equipos de red y posteriormente enviándolas desde un controlador.

Tras la realización de esta memoria, se pretende obtener un proyecto escalable, con opciones de continuidad y útil en un entorno empresarial real.

1.4. ORGANIZACIÓN DE LA MEMORIA

La memoria se compone de los siguientes apartados:

1. En este primer bloque, se pondrá en situación al lector sobre el marco del trabajo y sus objetivos. Se describe brevemente la problemática abordada y de qué manera se va a afrontar.
2. En esta sección se procede a describir las herramientas, centrándonos en entornos de virtualización y máquinas virtuales, junto con los programas (software) usados para la realización de este Trabajo Fin de Grado.
3. En este tercer capítulo, se describen los escenarios de red sobre los cuales se ha trabajado y las pruebas que se han realizado sobre cada uno de ellos.
4. En este capítulo, se recogen las conclusiones obtenidas y se presentan ideas para la continuación de este Trabajo.
5. Bibliografía y referencias.
6. Posteriormente en los Anexos, se describen detalladamente las configuraciones realizadas sobre las máquinas en cada escenario.

2. HERRAMIENTAS

En este segundo capítulo, se procede a describir las herramientas (entornos de virtualización, máquinas virtuales y software) utilizadas para el desarrollo de este Trabajo Fin de Grado. Se pretende así introducir y dar un contexto de sus principales características y el motivo por el cual han sido elegidas, ya que el abanico de posibilidades es muy amplio y se probaron numerosas soluciones diferentes antes del desarrollo definitivo.

2.1. ENTORNOS DE VIRTUALIZACIÓN

"El termino virtualización se refiere a la creación a través de software de una versión virtual de algún recurso tecnológico, como puede ser una plataforma de hardware, un sistema operativo, un dispositivo de almacenamiento u otros recursos de red." (Turban, E; King, D; Lee, J; Viehland, D (2008). Electronic Commerce A Managerial Perspective) [5]

Como se expuso en el primer capítulo, finalmente nos decidimos a montar nuestros escenarios de pruebas en un entorno virtualizado por seguridad, simplicidad y la posibilidad de tener el control total de la red. Tras varias pruebas de rendimiento las herramientas que mejor se adaptaban a nuestras necesidades y por tanto las elegidas para la virtualización son:

2.1.1. GNS3

GNS3 (*Graphic Network Simulation*) es un simulador gráfico de redes lanzado en 2008, que permite diseñar topologías de red complejas y poner en marcha simulaciones sobre ellas. Los usuarios tendrán la posibilidad de poder escoger cada uno de los elementos que llegarán a formar parte de una red informática permitiendo la combinación de dispositivos tanto reales como virtuales.

Es uno de los mejores simuladores gráficos de redes que podemos encontrar actualmente, tanto para estudiantes, como para el mundo empresarial y pruebas de laboratorio. La principal característica de GNS3 es que es multiplataforma, podremos usarlo tanto en Microsoft Windows, Linux como en Mac OS X. Además, esta herramienta es gratuita, debido a que es un software *open source*, sigue en constante desarrollo por la comunidad. Su código fuente se puede encontrar en GitHub. [6]

Originalmente solo emulaba dispositivos Cisco que usaban un software llamado Dynamips. Ahora ha evolucionado y admite dispositivos de múltiples proveedores de red, incluidos conmutadores virtuales Cisco, Cisco ASA, Brocade vRouters, instancias Docker, HPE VSR, múltiples dispositivos Linux y muchos más. [7]

En este Trabajo Fin de Grado, se ha utilizado la versión GNS3 2.1.14. Esta herramienta nos ha permitido crear y virtualizar los escenarios de red deseados, a la vez que emular distintas máquinas virtuales en nuestro equipo real. Además, nos da la posibilidad de poder ejecutar los servidores para nuestras máquinas en varias ubicaciones. Al mismo tiempo que podemos ejecutar máquinas en nuestro equipo local, también tenemos la opción de hacerlo sobre otro software de virtualización (VMware, VirtualBox, QEMU…) e incluso sobre un servidor remoto. Durante este Trabajo, hemos aprendido a configurar todas estas opciones disponibles, para después usar la más adecuada en cada máquina y situación.

2.1.2. QEMU

QEMU es un emulador genérico y de código abierto. Cuando se utiliza como un emulador de máquinas virtuales, QEMU puede correr sistemas operativos y programas hechos para una máquina en particular, por ejemplo, una placa ARM, puede correr en un PC x86. Esta emulación se basa en la traducción dinámica de binarios (conversión del código binario de la arquitectura fuente en código entendible por la arquitectura huésped). [8]

Además, QEMU tiene capacidades de virtualización dentro de un sistema operativo, ya sea GNU/Linux, Windows, o cualquiera de los sistemas operativos admitidos. El objetivo principal es emular un sistema operativo dentro de otro sin tener que particionar el disco duro. [9]

En este Trabajo Fin de Grado se utiliza QEMU para la virtualización de los sistemas operativos y distribuciones Linux que se van a utilizar en los diferentes escenarios. De este modo, evitamos problemas de compatibilidad con nuestro equipo real. Con esta emulación, disponemos de los sistemas operativos que tendríamos instalados en diferentes máquinas, corriendo virtualmente y de manera simultánea en nuestro equipo local.

El único inconveniente para el uso de QEMU es la compatibilidad con Windows, ya que hasta ahora su desempeño no ha sido el esperado, tiene numerosas cadencias en cuanto a rendimiento. Por ello procedemos a ejecutarlo sobre equipos Linux, en los cuales el rendimiento es bueno. Una vez solventado este problema, mediante el uso de una máquina virtual de VMware, QEMU es mucho más ligero además de tener disponibles numerosas máquinas ya creadas para su directa utilización.

2.1.3. VMWARE WORKSTATION 12 PLAYER

VMware Workstation Player es una aplicación de virtualización de escritorios con la que poder ejecutar una máquina virtual en un PC con Windows o Linux. Permite ejecutar un segundo sistema operativo aislado en un mismo PC. Utiliza *VMware vSphere Hypervisor* para ofrecer una solución de virtualización local sencilla pero estable y consolidada, que tiene muchos usos, por ejemplo, como herramienta personal de aprendizaje o como herramienta comercial para optimizar la ejecución de un escritorio empresarial en un dispositivo personal.

Una vez instalado, VMware Workstation Player permite instalar otros sistemas operativos y ejecutarlos como máquinas virtuales dentro de una ventana independiente. Incluye funciones que permiten a los usuarios crear y configurar sus propias máquinas virtuales para lograr el rendimiento óptimo y acceder a todos los dispositivos conectados a su PC. [10]

En este Trabajo Fin de Grado, se ha utilizado la versión 12.0.0 de VMware Workstation Player y se ha creado una máquina virtual con sistema operativo Ubuntu 64-bit en la versión 14.04.5. Además, en ella se ha instalado GNS3, del que se ha hablado anteriormente. Gracias a esta instalación, GNS3 nos ha permitido usar esta máquina virtual como servidor para los diferentes nodos de nuestro escenario. GNS3 también permite el uso de VirtualBox, otra aplicación de virtualización con la que también estamos familiarizados, pero se recomienda el uso de VMware, y el rendimiento es significativamente mejor. Esta unión de sistema Linux con la utilización de QEMU, nos permite ejecutar las virtualizaciones con mayor velocidad y fiabilidad.

2.2. MÁQUINAS A UTILIZAR

En GNS3 tenemos disponibles numerosas *appliances*[1] por defecto, pero además podemos descargar e instalar tantas como queramos o necesitemos, para tener total libertad a la hora de elegir los equipos a utilizar. Para este Trabajo Fin de Grado, por motivos de sencillez, ligereza y predilección por el software libre, se han elegido las siguientes:

2.2.1. TINY CORE LINUX

Es un sistema operativo minimalista con núcleo Linux. Lo más destacable de la distribución es su reducido tamaño (16 megabytes) y su minimalismo, a pesar de proporcionar una interfaz gráfica completa. Posee un repositorio que contiene más de 3.200 extensiones (aplicaciones, bibliotecas, etc.), los cuales proveen funciones adicionales. Tiny Core Linux es un software libre de código abierto, bajo licencia GNU.

[1] Una *appliance* podría describirse como una plantilla de una máquina virtual que permite hacer réplicas de esta, ejecutando todas ellas sobre un mismo servidor. Con el uso de *appliances* evitamos el tener que instalar una máquina virtual como servidor para cada máquina que queramos utilizar.

Los desarrolladores lo describen como "un sistema operativo de escritorio gráfico ultraligero capaz de arrancar desde un CD-ROM, USB, o desde una instalación sencilla desde un disco duro". [11]

En su configuración, Tiny Core Linux se carga completamente en la memoria RAM. Este tiene acceso a la red, incluye un servidor “ssh”, entorno de ventanas, panel de control, etc. Proporciona así un sistema operativo actualizado y estable para el usuario promedio, con un fuerte enfoque en la facilidad de uso y de instalación del sistema, además de su ligereza.

En este Trabajo Fin de Grado se utilizan de manera general máquinas con sistema operativo Linux, ejecutadas sobre una máquina virtual de VMware. Se ha elegido esta distribución para los nodos finales que realizan las funciones de PC, con el objetivo de generar y recibir tráfico. Como se mencionaba en la introducción, la elección de esta distribución es debido a su ligereza. El uso de máquinas simples, con recursos limitados, nos hace entender que posteriormente podremos escalar el proyecto. Además, de esta distribución se destaca del resto por poseer un entorno gráfico, lo que hará más visual el resultado final.

2.2.2. MICRO CORE LINUX

Esta es una variante oficial de Tiny Core Linux. Se desarrolla desde la misma base Linux y con las mismas premisas de ser simple y ligero (pesa unos 11 megabytes). Cabe destacar, que esta distribución es más pequeña y no tiene entorno gráfico, aunque se puede añadir posteriormente mediante extensiones.

Adicionalmente, existen otras muchas distribuciones, unas oficiales y otras creadas por la comunidad. Cada una posee unas determinadas características y objetivos finales distintos, por eso hay que saber cuál es el propósito de uso y así poder elegir la que mas se adapte a nuestras necesidades.

Como se está mencionando desde el inicio, en este Trabajo Fin de Grado se busca utilizar máquinas simples. Se ha elegido esta distribución para los nodos centrales que realizarán las funciones de *switch*. El motivo principal es su ligereza, mayor incluso que la de Tiny Core Linux. En este caso, no es necesario disponer de un entorno gráfico, por lo que en este sentido ahorraremos recursos. Nos bastará con la consola de comandos para aplicar configuraciones y realizar pruebas de comunicación, como si se tratase de un *switch* real. Sobre estos Micro Core Linux, podemos instalar el software necesario ya que siguen siendo equipos con base Linux. En nuestro caso se ha instalado el software Open vSwitch necesario para dotar a estos quipos con las funciones de *switch* virtual. Adicionalmente, a uno de los equipos se le dotará con funciones de *router*, y a pesar de la simplicidad de los equipos, soportará sin problemas esta tarea. Todos ellos serán accesibles por el controlador a través de “ssh”. Por tanto, son equipos que destacan por su ligereza y versatilidad.

2.2.3. UBUNTU

Ubuntu es un sistema operativo de código abierto para computadores. Es una distribución de Linux basada en la arquitectura de Debian. Incluye principalmente software distribuido bajo una licencia libre o de código abierto, haciendo excepciones en el caso de algunos controladores privativos. Actualmente corre en computadores de escritorio y servidores, en arquitecturas Intel, AMD y ARM.

Su primer lanzamiento fue en octubre de 2004, siendo una bifurcación del código base del proyecto Debian. El objetivo inicial era hacer una distribución más fácil de usar y entender para los usuarios finales, corregir varios errores y hacer más sencillas algunas tareas como la gestión de programas. A partir de entonces, ha seguido evolucionando y sacando versiones cada seis meses y manteniendo el soporte de cada versión durante dos años. Además, esta evolución les ha permitido crear versiones para móviles, televisiones, etc. [12]

En este Trabajo Fin de Grado, se ha elegido utilizar Ubuntu como nodo central, de mayor potencia para hacer las veces de controlador de la red. Se ha elegido la versión 16.04 LTS por ser *Long Term Support,* esto quiere decir que tiene cinco años de soporte en lugar de los dos que es el estándar. Esta versión es la actual LTS con más recorrido, considerándose una de las versiones más estables hoy en día. Adicionalmente, hay que destacar que Ubuntu es una distribución libre y muy utilizada en las tareas de *networking*. También es una de las versiones recomendadas por el propio GNS3 en: https://gns3.com/marketplace/appliance/ubuntu-with-gui [13]

Se ha instalado como cualquier otra *appliance* en GNS3 y solamente se usará como sistema operativo para el controlador debido a su peso y requisitos técnico. Con el uso de este equipo queremos mostrar la posibilidad de utilizar un único equipo, con mayor peso y capacidad de procesado, como controlador de varias redes al mismo tiempo, que, aunque no será nuestro caso, conviene tenerlo en cuenta para las posibles futuras etapas del proyecto.

Por los motivos descritos, se ha elegido Ubuntu, pero en el equipo de investigación en el cual esta enmarcado este Trabajo, se ha conseguido configurar un controlador incluso hasta en una Raspberry Pi con el sistema operativo Raspbian. De esta forma queda demostrado que pueden utilizarse equipos simples para las tareas de control, aunque en nuestro caso no es necesario minimizar y optimizar tanto.

2.3. HERRAMIENTAS SOFTWARE

Las máquinas elegidas, junto con la posibilidad de conectarlas a Internet, nos brindan la oportunidad de poder moldear los equipos a nuestro antojo. Esto nos permite instalar softwares en las máquinas, para alcanzar los objetivos o tener interfaces gráficas que hagan más visual el Trabajo. Hay que destacar que se ha experimentado previamente con numerosas herramientas, pero finalmente las elegidas son las que mejor encajan siendo acordes con la capacidad de las máquinas ligeras anteriormente elegidas. Algunos de los softwares que hemos instalado son los siguientes:

2.3.1. GENERADOR DE TRÁFICO – OSTINATO

Es un software *open source* que permite generar tráfico para posteriormente analizarlo. Cuenta con una interfaz gráfica bastante sencilla que nos permite crear y enviar paquetes en diferentes transmisiones con diferentes protocolos (Ethernet, 802.3, LLC SNAP, ARP, IPv4, IPv6, TCP, UDP, ICMPv4, ICMPv6, IGMP, MLD, HTTP, SIP…) a diferentes velocidades.

Ostinato tiene como objetivo proporcionar un generador de tráfico y una herramienta de prueba de red para ingenieros y desarrolladores de redes. Con esta herramienta pueden hacer mejor su trabajo, tanto con pruebas funcionales, como de rendimiento para mejorar la calidad de los productos de red. Aunque es muy útil para experimentar en situaciones realistas, hay que tener en cuenta que Ostinato, por motivos de seguridad, no admite conexiones TCP orientadas a conexiones con estado ni generación de tráfico falso a sitios web.

En este Trabajo Fin de Grado, se ha utilizado Ostinato para generar diferentes tipos de tráfico en diferentes enlaces y así poder analizar el mismo escenario en diferentes situaciones.

Además de este software especifico con el que podemos generar tráfico de diferentes protocolos, a la hora de comprobar la conexión entre 2 máquinas, se podrá realizar de manera más ágil con la herramienta sobradamente conocida, “ping”, que se comunica con el destino mediante paquetes ARP e ICMP.

2.3.2. ANALIZADOR DE TRÁFICO – WIRESHARK

Wireshark es un analizador de protocolos de software libre, que puede ejecutarse sobre la mayoría de los sistemas operativos, incluso los portables. Se utiliza para realizar análisis y solucionar problemas en redes de comunicaciones, para desarrollo de software y protocolos, además de como una herramienta didáctica. Para la captura de tráfico utiliza “tcpdump” o “dumpcap” y además añade una interfaz gráfica con muchas opciones de organización y filtrado de información que hace que su uso sea sencillo e intuitivo.

Este analizador de tráfico nos permite dos modos para el análisis. El primero, permite ver todo el tráfico que pasa a través de una red de manera *online* o en tiempo real. La segunda, de manera *offline*, permite examinar datos de un archivo de captura previa guardada. Mediante estos dos modos de captura se puede analizar la información del tráfico, a través de los detalles de cada paquete. Wireshark incluye un completo lenguaje para filtrar lo que queremos ver y la habilidad de mostrar el flujo reconstruido de una sesión de TCP.

Con toda esta información y ventajas, hay que destacar qué Wireshark no es un sistema de detección de intrusos. No advierte cuando un intruso hace algo «extraño» en la red en la que está capturando el tráfico. No manipula, ni envía datos, solamente inspecciona los paquetes de la red, para qué alguien pueda analizar los datos posteriormente y decidir tomar acciones si sospecha que hay algo extraño en la red. Por este motivo, puede considerarse como un analizador de protocolos pasivo (no actúa sobre la red). [14]

En nuestro caso será utilizado para capturar el tráfico generado en las interfaces adecuadas, para su posterior análisis.

2.3.3. OPEN VSWITCH

Open vSwitch es un software multicapa de código abierto, diseñado para ser utilizado como un *switch* virtual en cualquier entorno. Su objetivo es la implementación de una plataforma que soporte interfaces de gestión estándar y, además, permita las funciones de *forwarding* de forma programable. Es utilizado para reenviar el tráfico entre diferentes máquinas virtuales (VMs) en el mismo host físico y también es capaz de reenviar el tráfico entre las máquinas virtuales y la red física. Todo lo cual, nos permite construir redes SDN.

A lo largo del tiempo, desde que comenzó su desarrollo en 2009 ha ido actualizándose y ampliando su compatibilidad. La versión actual de Open vSwitch es posee diferentes características, algunas de las destacadas para este Trabajo Fin de Grado pueden ser: OpenFlow, herramientas para el monitoreo de enlaces, STP, QoS, soporte para protocolos de túneles y firewall, etc.

Open vSwitch puede ser portado a diferentes plataformas de hardware y diferentes sistemas operativos. Además, soporta la administración remota de una manera que hace que así sea más fácil para los desarrolladores de las plataformas de virtualización.

La configuración del Open vSwitch es controlada por un esquema de base de datos organizada en varias tablas. Algunas herramientas que proporciona Open vSwitch:

- Ovs-vsctl: es una utilidad para crear y administrar toda la configuración referente a los *bridge*, puertos e interfaces del *switch*.
- Ovs-ofctl: es una utilidad para consultar y manejar *switch* y controladores.
- Ovs-testcontroller: es un controlador OpenFlow básico útil para pruebas.

Esta información se puede ampliar en: http://docs.openvswitch.org/en/latest/ [15]

Para su mejor uso y comprensión, los comandos utilizados, en este Trabajo Fin de Grado estarán reunidos y serán explicados en los Anexos.

Por todo lo descrito anteriormente, este será nuestra herramienta principal. Con Open vSwitch construiremos el *switch* virtual sobre nuestros equipos Micro Core Linux. Esto nos permitirá que nuestra red pase a estar definida por software y dotarla ahora de mayor funcionalidad que la que tendría una red con *switch* normales. Con las redes formadas con *switch* de capa 2 se construye la tabla de conmutación en base a las direcciones MAC origen. Ahora, con las redes definidas por software, podemos conformar la tabla de conmutación en base a una n-tupla. Esto quiere decir que no solo podemos analizar y almacenar las direcciones MAC origen, sino que podemos conformar con libertad la conmutación de los flujos de tráfico deseados utilizando, por ejemplo, el tipo de tráfico, las direcciones IP, la pertenencia a una determinada VLAN, etc. Indicando los puertos de entrada y salida que queramos para la conmutación de ese flujo. Esto nos abre las puertas a una gestión más completa del tráfico.

2.3.4. SSH

SSH (*Secure Shell*) es un protocolo que posibilita el acceso y la administración de un servidor remoto. Además, da nombre al programa que permite su implementación. A diferencia de otros protocolos como HTTP o FTP, SSH establece conexiones seguras entre los dos sistemas. Esto se produce recurriendo a la llamada arquitectura cliente/servidor.

Como ya ocurrió con el protocolo TCP, SSH nació para sustituir a otros protocolos. Por su antigüedad, estos protocolos resultaban inseguros para la conexión o el intercambio de datos en Internet. En este sentido, el antecedente de SSH es Telnet (*Teleccomunication Network*). Un protocolo de red en modo terminal para acceder a una máquina remota desarrollado en 1969. La diferencia entre ambos es que SSH añade seguridad mediante cifrado. En cambio, con Telnet toda la información viaja por internet en forma de texto plano haciéndola fácilmente accesible. [16]

En este Trabajo Fin de Grado, este protocolo se utiliza para poder conectar remotamente el controlador a los diferentes *switch.* Esto nos permite enviar todas las órdenes a los diferentes equipos desde un mismo lugar centralizado, a la vez que crear scripts en el controlador para ser ejecutados directamente en los nodos de la red. Toda la configuración realizada se encuentra redactada en el Anexo 2.

3. ESCENARIOS DE PRUEBAS

En este tercer capítulo, se describen los diferentes escenarios de red montados, con imágenes y explicaciones de cómo están configurados y por qué se ha decidido mostrar cada uno de ellos.

Durante la realización de este Trabajo Fin de Grado, se han montado diferentes escenarios y se han realizado numerosas pruebas con el fin de cumplir los objetivos propuestos. A continuación, se presentan los principales, que son los que se consideran que han tenido mayor relevancia y pueden aportar más información al lector.

3.1. ESCENARIO 1 – CONEXIÓN A INTERNET

Este primer escenario, destaca por ser el nexo entre las redes físicas y el propio escenario virtualizado. Se cree que es apropiado e importante destacarlo, ya que, a través de él y sus configuraciones, si es necesario, podemos comunicar cualquier nodo de los futuros escenarios con Internet.

El punto clave de este escenario es el nodo "Cloud". Es un nodo predefinido que permite la salida a internet de otras máquinas, mientras se ejecuta en nuestro equipo local. Este nodo se conecta a un equipo cualquiera, con funciones de "iptables" y "NAT" (*Network Address Translation*), que será el que haga las veces de *router* dentro de esta red. En este caso, está conectado a un nodo "Ostinato0.9", que es un Tiny Linux, elegido por simplicidad y unificar el tipo de máquinas a utilizar, aunque podría valer cualquier otro cómo un VPC u otro Linux. Con este proceso, se tiene conectividad dentro de los escenarios de red en GNS3. Una buena práctica será conectar este equipo, al que hemos dotado con funciones de *router*, a un *switch* y conectar a él toda la red, de manera ordenada, para que todos ellos tengan acceso a internet. Bastará conectar a este cualquier equipo de la red, con las direcciones IP adecuadas para poder tener conexión a internet en todos ellos. A ésta la llamaremos la red de gestión de nuestros escenarios. En el Anexo 1 se pueden consultar los detalles del direccionamiento IP en las máquinas.

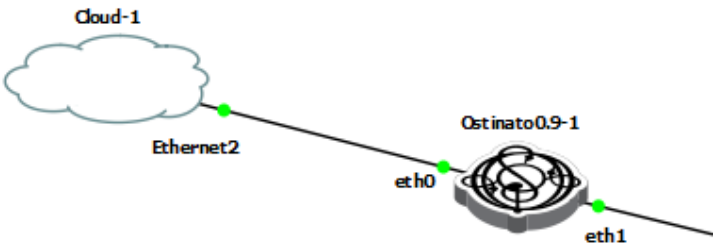


Ilustración 1: Escenario 1 - Conexión a internet

La configuración de este escenario es relativamente sencilla, si se sabe que es lo que se debe hacer exactamente. Lo primerio es configurar adecuadamente las direcciones IP, máscaras y *gateway* dentro de la máquina que va a hacer de *router*. Después se deben habilitar las "iptables" con el siguiente comando:

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Ilustración 2: Habilitación de iptables en Linux

De esta forma el equipo conectado al nodo *"cloud"* [2] que podía salir a internet, ahora hará "NAT" y permitirá salir a Internet a los demás equipos que le conectemos. Posteriormente, se podrá añadir por ejemplo un *switch*, como se ha comentado antes, para mejorar la organización y permitir a más equipos salir a Internet.

En las siguientes imágenes podemos comprobar mediante la herramienta ping como podemos conectarnos a Internet, por ejemplo, con google.com.

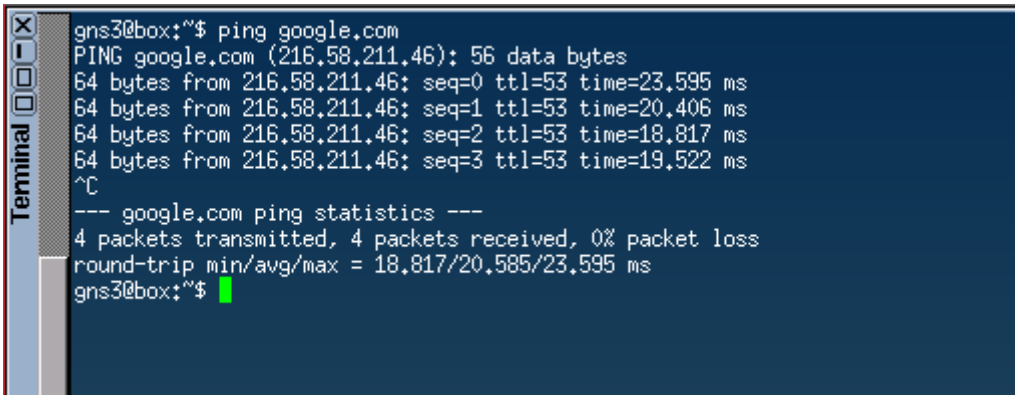


Ilustración 3: Ejemplo ping google.com

82 24.844022	192.168.0.35	212.166.132.96	DNS	70	Standard query 0x01c7 A google.com
83 24.844237	192.168.0.35	212.166.132.96	DNS	70	Standard query 0x01c7 A google.com
84 24.845021	192.168.0.35	212.166.132.96	DNS	70	Standard query 0xed74 AAAA google.com
85 24.845194	192.168.0.35	212.166.132.96	DNS	70	Standard query 0xed74 AAAA google.com
86 24.866877	212.166.132.96	192.168.0.35	DNS	86	Standard query response 0x01c7 A google.com A 216.58.211.46
87 24.867667	212.166.132.96	192.168.0.35	DNS	98	Standard query response 0xed74 AAAA google.com AAAA 2a00:1450:4003:803::200e
88 24.870238	192.168.0.35	216.58.211.46	ICMP	98	Echo (ping) request id=0x8119, seq=0/0, ttl=64 (no response found!)
89 24.870445	192.168.0.35	216.58.211.46	ICMP	98	Echo (ping) request id=0x8119, seq=0/0, ttl=64 (reply in 90)
90 24.890861	216.58.211.46	192.168.0.35	ICMP	98	Echo (ping) reply id=0x8119, seq=0/0, ttl=53 (request in 89)
91 24.935280	0c:fe:89:c9:1c:00	Sagemcom_c0:20:34	ARP	60	Who has 192.168.0.1? Tell 192.168.0.35
92 24.935629	0c:fe:89:c9:1c:00	Sagemcom_c0:20:34	ARP	60	Who has 192.168.0.1? Tell 192.168.0.35
93 24.937013	Sagemcom_c0:20:34	0c:fe:89:c9:1c:00	ARP	60	192.168.0.1 is at e8:ad:a6:c0:20:34
94 25.872285	192.168.0.35	216.58.211.46	ICMP	98	Echo (ping) request id=0x8119, seq=1/256, ttl=64 (no response found!)
95 25.872563	192.168.0.35	216.58.211.46	ICMP	98	Echo (ping) request id=0x8119, seq=1/256, ttl=64 (reply in 96)
96 25.890856	216.58.211.46	192.168.0.35	ICMP	98	Echo (ping) reply id=0x8119, seq=1/256, ttl=53 (request in 95)

Ilustración 4: Ejemplo ping google.com – Captura Wireshark

Con este primer escenario conseguimos enlazar nuestras máquinas virtuales con Internet. Esto es de gran utilidad ya que, mediante los comandos adecuados, nos permitirá instalar cualquier software que deseemos en cualquiera de los nodos. Por ejemplo, Open vSwitch, que es el software fundamental utilizado para el desarrollo de este Trabajo Fin de Grado. Adicionalmente, podríamos enviar y recibir tráfico del exterior de la red si así lo deseásemos.

[2] Como detalle importante, la interfaz de red en nuestro equipo local debe tener un nombre distinto de "Ethernet", por ejemplo "Ethernet2" o cualquier otro, sino no funcionará la conexión.

3.2. ESCENARIO 2 – *LEARNING SWITCH*

Este es un escenario donde se pretende mostrar las capacidades de configuración que tiene Open vSwitch, haciéndolo funcionar como un *switch* normal o *learning switch.* Además, se introduce la figura del controlador, en este caso un equipo "Ubuntu", que permitirá realizar las tareas de control en la red. Este escenario es fundamental, ya que, aparte de mostrar una visión general de un escenario con conexión al exterior y un controlador, demuestra que, si instalamos el software Open vSwitch en cualquier equipo y no configuramos nada, este no realizará ninguna función de retransmisión de tráfico, debido a que, en el fondo simplemente será una maquina Linux con un software sin configurar. De esta manera tenemos el control total sobre nuestros escenarios. En este caso queremos configurar, mediante las ordenes correspondientes, el *switch* para que funcione como lo haría uno en la realidad y tenga las funciones que debemos poner a prueba y analizar.

Para entender mejor nuestro objetivo, podemos observar la siguiente ilustración, en la que se representa el nodo "ovs", que hará de *switch* en nuestros escenarios. En él, se ve un mapeado entre los puertos físicos del nodo, con un módulo que representa el software instalado llamado Open vSwitch, con el que vamos a dotar de todas las funciones de *switch* virtual al equipo. A estos enlaces los llamaremos "externos". [3] Además, vemos otro módulo llamado "control", que nos permite darle todas las instrucciones para gestionar los recursos del software para que funcione como nosotros le indiquemos, creando así una **red para el control**.

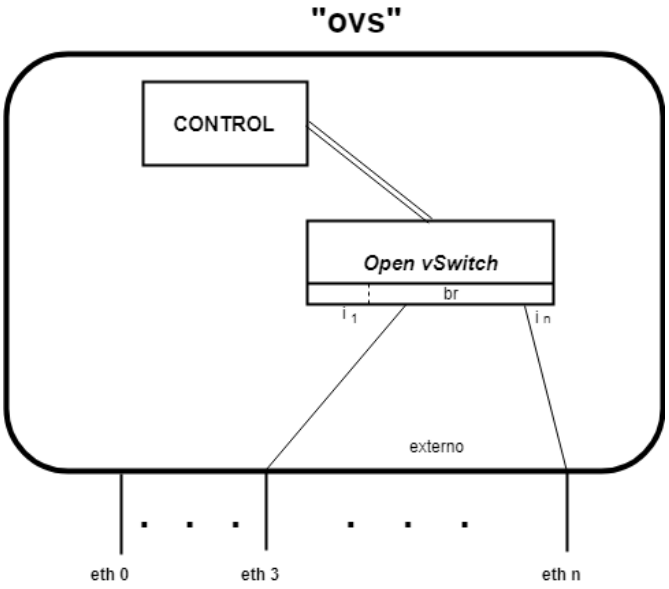


Ilustración 5: Representación de switch con Open vSwitch

[3] En la Ilustración 5 podemos ver como las interfaces físicas "eth" no tienen por qué coincidir con los puertos en el "ovs", esto afectará a la configuración, cuando se incluyan flujos de datos.

Para el montaje de este escenario, incluimos tres nodos “Ostinato” que hacen las veces de PC final, un nodo “ovs” que será en el que configuremos nuestro *switch* virtual para que funcione como si se tratase de un *switch* real, alrededor del cual se creará la red de datos y un equipo “Ubuntu”, que hará las veces de controlador de la red. Este podrá conectarse mediante “ssh” al *switch* de la red para poder ejecutar scripts y comandos de configuración de flujos. Los detalles de la configuración del servicio “ssh” se pueden consultar en el Anexo 2.

Las tres máquinas “Ostinato” estarán configurados en una misma red, para realizar las pruebas que exponemos posteriormente y conformar así la **red de datos**. Al nodo “ovs” se le configurará una dirección para que se pueda comunicar con el controlador y comunicarse en la **red de control**. Finalmente, todos los equipos de la red estarán conectados a un *switch*, que conectado al equipo que hace las funciones de *router*, permitirá la salida a internet de estos utilizando **la red de gestión**.

De este modo, ya tendremos definidas las 3 redes necesarias para el análisis de una red definida por software (control, gestión y datos) y con las que se va a realizar el análisis en este Trabajo Fin de Grado.

Finalmente, en este escenario, a los PC les hemos configurado sus correspondientes direcciones IP, para que estén dentro de la misma red (192.168.1.0/24) como se puede ver en la Ilustración 6. En el Anexo 1 se puede encontrar la descripción de cómo se configura el direccionamiento IP en las diferentes máquinas. Cada equipo está conectado a un interfaz del *switch* como si se tratase de un equipo físico. Posteriormente, se ha configurado el nodo “ovs” para que actúe como *switch* a nivel *ethernet*, y aprenda las rutas a seguir que nosotros le indiquemos para el reenvío de paquetes mediante las direcciones MAC en esta red. Se puede ver también las direcciones de las redes para la red de control.

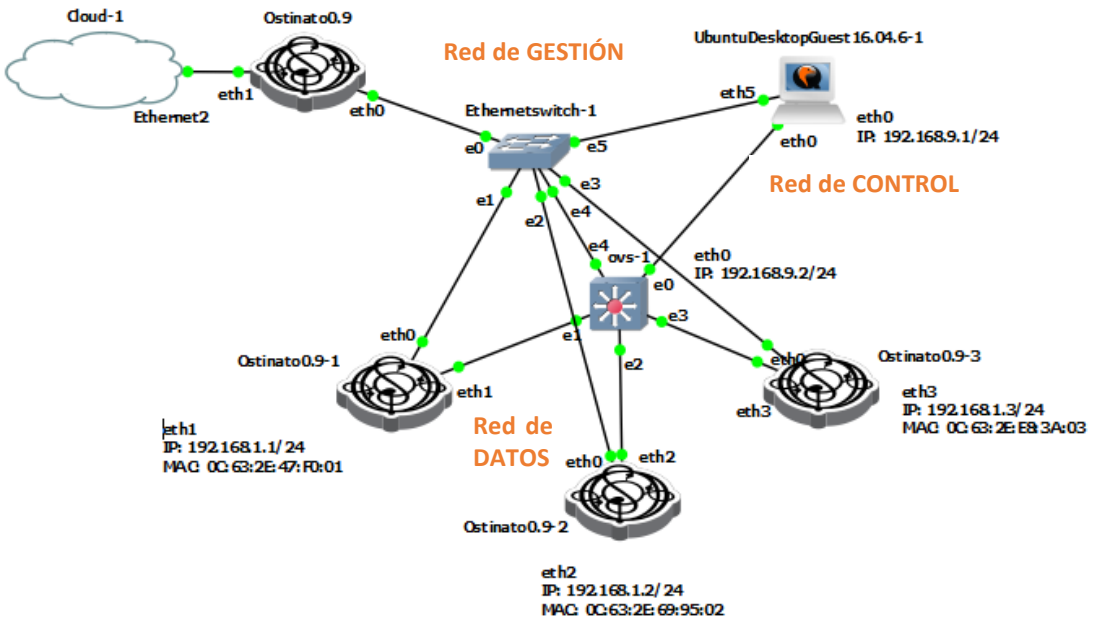


Ilustración 6: Escenario 2 - Learning switch completo

Como se puede observar, el esquema de la red queda bastante colapsado por la cantidad de equipos y uniones que lo componen. Por este motivo, para una mejor comprensión, se decide que, a partir de este momento, en las posteriores Ilustraciones donde se pueden ver los escenarios, solamente se mostrará la red de datos integrada por los equipos finales "Ostinato" y los *switch* "ovs". De esta manera se verá con mayor claridad el escenario, sabiendo siempre que existe la posibilidad de conectar los equipos a las redes de control y gestión. Por tanto, el escenario anterior quedaría de la siguiente manera:

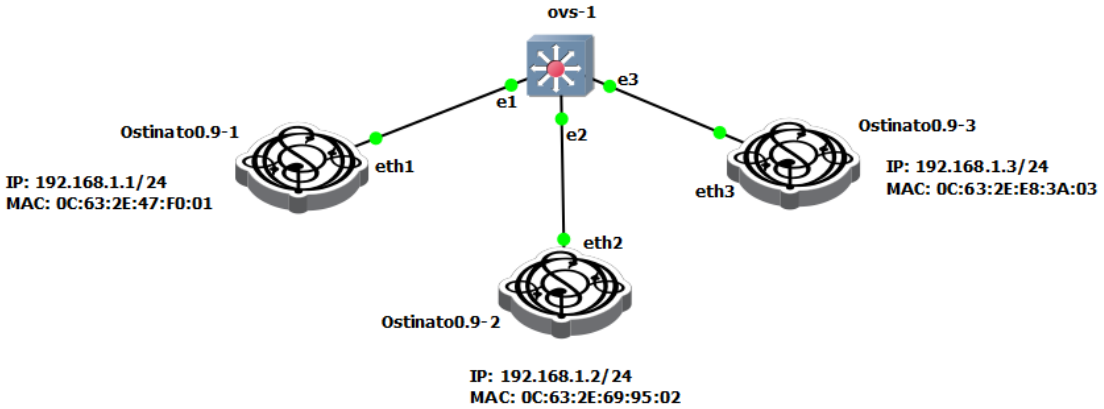


Ilustración 7: Escenario 2 - Learning switch simplificado

Para la configuración en el "ovs" se ha creado un *bridge* y se le han definido dentro las interfaces correspondientes donde se han conectado los PC. Después se han definido los flujos por los que tiene que reenviarse el tráfico de tal manera que los paquetes con dirección destino *broadcast* se reenvíen por todos los enlaces y los paquetes con dirección destino *unicast* (dirección MAC), se reenvíen solamente por el enlace correspondiente y no por los demás. De esta manera, conseguimos que no se replique el tráfico y los demás equipos de la red no vean el tráfico que es *unicast* y no está dirigido a ellos. El código con el que se han configurado estos flujos se describe en el Anexo 3.

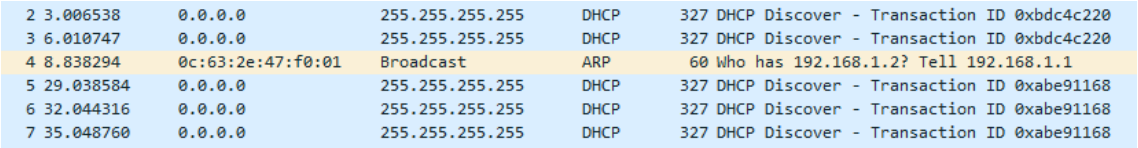
Mediante las siguientes capturas de Wireshark se puede observar cómo si se realiza un "ping" desde el "Ostinato0.9- 1" al "Ostinato0.9- 2", la trama ARP con dirección destino *broadcast* se podrá capturar en todos los enlaces. Pero las tramas *unicast*, tanto ARP cómo ICMP, solamente serán capturadas en los enlaces que comunican a los dos "Ostinato" involucrados en la comunicación.

Captura en el interfaz "eth2" del nodo "ovs-1":

6	35.051121	0.0.0.0	255.255.255.255	DHCP	327 DHCP Discover - Transaction ID 0xc27d1918
7	38.909252	0c:63:2e:47:f0:01	Broadcast	ARP	60 Who has 192.168.1.2? Tell 192.168.1.1
8	38.911097	0c:63:2e:69:95:02	0c:63:2e:47:f0:01	ARP	60 192.168.1.2 is at 0c:63:2e:69:95:02
9	38.913911	192.168.1.1	192.168.1.2	ICMP	98 Echo (ping) request id=0x1615, seq=0/0, ttl=64 (reply in 10)
10	38.915397	192.168.1.2	192.168.1.1	ICMP	98 Echo (ping) reply id=0x1615, seq=0/0, ttl=64 (request in 9)
11	39.925457	192.168.1.1	192.168.1.2	ICMP	98 Echo (ping) request id=0x1615, seq=1/256, ttl=64 (reply in 12)
12	39.926817	192.168.1.2	192.168.1.1	ICMP	98 Echo (ping) reply id=0x1615, seq=1/256, ttl=64 (request in 11)

Ilustración 8: Ping PC-1 a PC-2 captura en Interfaz "eth2" de "ovs-1"

Captura en el interfaz “eth3” del nodo “ovs-1”:



2	3.006538	0.0.0.0	255.255.255.255	DHCP	327 DHCP Discover - Transaction ID 0xbdc4c220
3	6.010747	0.0.0.0	255.255.255.255	DHCP	327 DHCP Discover - Transaction ID 0xbdc4c220
4	8.838294	0c:63:2e:47:f0:01	Broadcast	ARP	60 Who has 192.168.1.2? Tell 192.168.1.1
5	29.038584	0.0.0.0	255.255.255.255	DHCP	327 DHCP Discover - Transaction ID 0xabe91168
6	32.044316	0.0.0.0	255.255.255.255	DHCP	327 DHCP Discover - Transaction ID 0xabe91168
7	35.048760	0.0.0.0	255.255.255.255	DHCP	327 DHCP Discover - Transaction ID 0xabe91168

Ilustración 9: Ping PC-1 a PC-2 captura en Interfaz "eth3" de "ovs-1"

Con esta prueba, hemos verificado que somos capaces de controlar el tráfico a nivel de enlace de datos. Con las correspondientes configuraciones en los flujos, podemos diferenciar y controlar tráfico tanto si es *broadcast*, como si es *unicast*.

Cabe destacar que tanto en este, como en los demás escenarios en los que ya no aparece visualmente el controlador (aunque en realidad sí esté presente), podemos conectarnos desde él a los *switch* con “ssh”, explicado detalladamente en el Anexo 2. Una vez conectados, tendremos la opción de introducir las ordenes manualmente como en el *switch* directamente o escribirlas en un fichero de texto, copiarlas en el destino con “scp” y ejecutarlo remotamente. Estas opciones estarán presentes en los todos los escenarios, aunque no se detallen para no resultar repetitivos. Además, se ha comprobado que las órdenes de configuración pueden ejecutarse en cualquier momento, lo que permite ir modificando los flujos en función de las necesidades sin que haga falta el reinicio del sistema.

3.3. ESCENARIO 3 – *SWITCH* CON FUNCIONES DE *ROUTER*

Para dar continuidad a este Trabajo, el siguiente escenario creado es una ampliación del anterior. En este caso se desea añadir al mismo *switch* otro PC, pero que éste pertenezca a otra red. De este modo, un *switch* normal no permitiría la comunicación entre equipos de redes diferentes. Por ello, queremos dotar ahora a este *switch* con las funciones de un *router*, para que pueda reenviar tráfico entre redes y así pueda haber comunicación entre los equipos del escenario. Por tanto, mientras en el escenario anterior se estaban implementando flujos para controlar el tráfico a nivel de enlace de datos, ahora pasaríamos a controlar estos a nivel de red.

En un escenario teórico, lo más habitual sería tener dos aparatos físicos identificados como el *switch* y el *router.* Pero aquí, podemos configurarlo todo en el mismo dispositivo, para que toda la comunicación que estamos gestionando, se realice según las ordenes que introduzcamos en un solo nodo. De esta forma tendremos un *switch* de capa 3 o *multilayer switch*. Para verlo de manera más clara, extendiendo la representación del *swtich* de la Ilustración 5, podemos ver como mediante software, podemos añadir un módulo, que denominamos “red”, que ejercerá de *gateway* para ambas redes y permitirá la comunicación entre ellas. En este módulo de red, podremos mapear tanto con interfaces que se comuniquen directamente con los puertos del dispositivo de manera externa, como internas hacia el Open vSwitch para que puedan comunicarse nuestras redes.

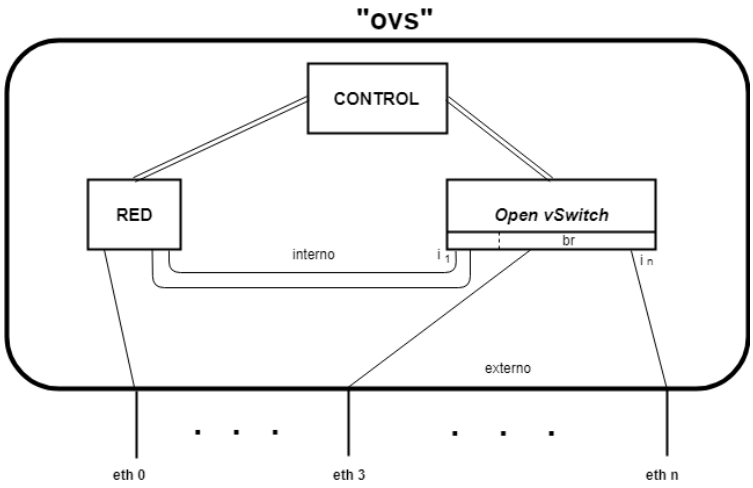


Ilustración 10: Representación de switch con Open vSwitch con funciones de router

Como hemos dicho, este escenario es una ampliación del anterior. Solamente hemos añadido otro PC más (situado a la derecha en la imagen) que pertenecerá a la red 192.168.2.0/24. Por lo tanto, tenemos tres equipos en una red y un equipo en otra, conectados mediante el mismo *switch.*

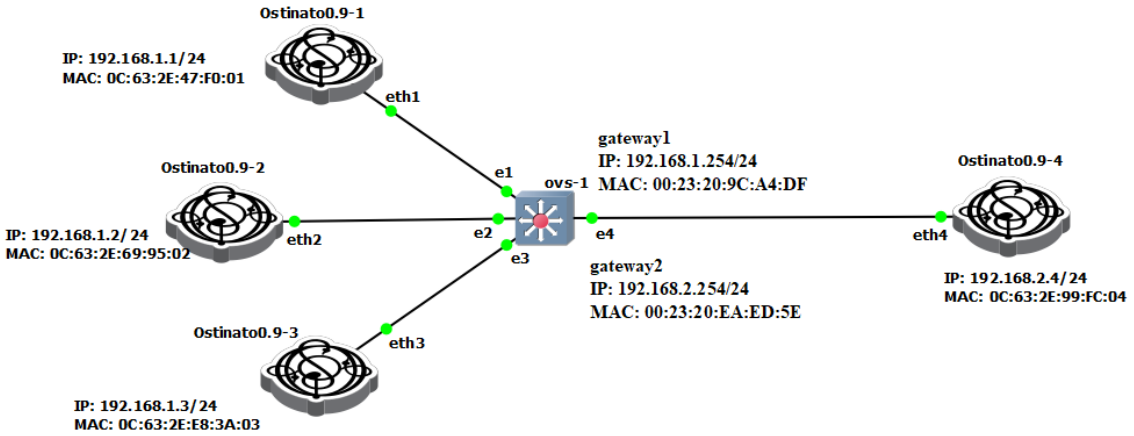


Ilustración 11: Escenario 3 - Switch con funciones de router

En la configuración de los PC de la red 192.168.1.0/24, no se ha modificado nada, simplemente se ha añadido un PC en otra red distinta, la 192.168.2.0/24 con sus correspondiente *Gateway.* La configuración de direccionamiento se puede consultar en el Anexo 1.

Después, en el *switch* se han configurado unas interfaces internas, para poder añadirles direcciones IP y que ejerzan de *gateway* para ambas redes. Tras esto, activaremos el *forwarding* para que el switch pueda reenviar tráficos de ambas redes. Finalmente, realizaremos la configuración adecuada de los flujos, diferenciando en este caso entre mensajes ARP, mediante direcciones MAC y paquetes ICMP mediante direcciones IP, demostrando así que la configuración en un Open vSwitch puede hacerse en varios niveles. Esta configuración completa puede verse en el Anexo 4.

En las siguientes capturas, observamos cómo mediante “ping” desde el “Ostinato0.9- 1” al “Ostinato0.9- 4”, situados en redes diferentes, éste es recibido por el “Ostinato0.9- 4” y en la red 192.168.1.0/24, por ejemplo, en el interfaz “eth2” del *switch,* solamente se ve el ARP *broadcast* del “Ostinato0.9- 1” preguntando por su g*ateway* para salir a la otra red.

Captura en el interfaz “eth4” del nodo “ovs-1”, perteneciente a la red destino:

174	1602.882634	0.0.0.0	255.255.255.255	DHCP	327 DHCP Discover - Transaction ID 0x28cfc545
175	1609.715199	NiciraNe_ea:ed:5e	Broadcast	ARP	42 Who has 192.168.2.4? Tell 192.168.2.254
176	1609.725247	0c:fe:47:67:06:04	NiciraNe_ea:ed:5e	ARP	60 192.168.2.4 is at 0c:fe:47:67:06:04
177	1609.726035	192.168.1.1	192.168.2.4	ICMP	98 Echo (ping) request id=0x9c15, seq=0/0, ttl=63 (reply in 178)
178	1609.726672	192.168.2.4	192.168.1.1	ICMP	98 Echo (ping) reply id=0x9c15, seq=0/0, ttl=64 (request in 177)
179	1610.715949	192.168.1.1	192.168.2.4	ICMP	98 Echo (ping) request id=0x9c15, seq=1/256, ttl=63 (reply in 180)
180	1610.716601	192.168.2.4	192.168.1.1	ICMP	98 Echo (ping) reply id=0x9c15, seq=1/256, ttl=64 (request in 179)
181	1611.717673	192.168.1.1	192.168.2.4	ICMP	98 Echo (ping) request id=0x9c15, seq=2/512, ttl=63 (reply in 182)

Ilustración 12: Ping PC-1 a PC-4 capturado en "eth4" del "ovs-1"

Captura en el interfaz “eth2” del nodo “ovs-1”, perteneciente a la red origen:

179	1570.904536	0.0.0.0	255.255.255.255	DHCP	327 DHCP Discover - Transaction ID 0x4995955e
180	1573.908444	0.0.0.0	255.255.255.255	DHCP	327 DHCP Discover - Transaction ID 0x4995955e
181	1582.751537	0c:fe:47:92:7a:01	Broadcast	ARP	60 Who has 192.168.1.254? Tell 192.168.1.1
182	1596.913217	0.0.0.0	255.255.255.255	DHCP	327 DHCP Discover - Transaction ID 0x88c76443
183	1599.919666	0.0.0.0	255.255.255.255	DHCP	327 DHCP Discover - Transaction ID 0x88c76443
184	1602.924865	0.0.0.0	255.255.255.255	DHCP	327 DHCP Discover - Transaction ID 0x88c76443

Ilustración 13: Ping PC-1 a PC-4 capturado en "eth2" del "ovs-1"

Realizando un “ping” interno en la red 192.158.1.0/24 desde el “Ostinato0.9- 1” al “Ostinato0.9- 2”, vemos como la comunicación existe y el “Ostinato0.9- 4” que pertenece a una red diferente no ve nada de este tráfico.

Captura en el interfaz “eth2” del nodo “ovs-1”, perteneciente a la red origen y destino:

194	1713.062411	0.0.0.0	255.255.255.255	DHCP	327 DHCP Discover - Transaction ID 0xc74ec957
195	1714.214619	0c:fe:47:92:7a:01	Broadcast	ARP	60 Who has 192.168.1.2? Tell 192.168.1.1
196	1714.215755	0c:fe:47:c6:cc:02	0c:fe:47:92:7a:01	ARP	60 192.168.1.2 is at 0c:fe:47:c6:cc:02
197	1714.228685	192.168.1.1	192.168.1.2	ICMP	98 Echo (ping) request id=0x9d15, seq=0/0, ttl=64 (reply in 198)
198	1714.230237	192.168.1.2	192.168.1.1	ICMP	98 Echo (ping) reply id=0x9d15, seq=0/0, ttl=64 (request in 197)
199	1715.221543	192.168.1.1	192.168.1.2	ICMP	98 Echo (ping) request id=0x9d15, seq=1/256, ttl=64 (reply in 200)
200	1715.223059	192.168.1.2	192.168.1.1	ICMP	98 Echo (ping) reply id=0x9d15, seq=1/256, ttl=64 (request in 199)

Ilustración 14: Ping PC-1 a PC-2 capturado en "eth2" del "ovs-1"

Captura en el interfaz “eth4” del nodo “ovs-1”:

194	1660.956480	0.0.0.0	255.255.255.255	DHCP	327 DHCP Discover - Transaction ID 0x74855649
195	1683.966771	0.0.0.0	255.255.255.255	DHCP	327 DHCP Discover - Transaction ID 0xf7840d14
196	1686.972904	0.0.0.0	255.255.255.255	DHCP	327 DHCP Discover - Transaction ID 0xf7840d14
197	1689.976654	0.0.0.0	255.255.255.255	DHCP	327 DHCP Discover - Transaction ID 0xf7840d14
198	1713.004287	0.0.0.0	255.255.255.255	DHCP	327 DHCP Discover - Transaction ID 0x1b851338
199	1716.007570	0.0.0.0	255.255.255.255	DHCP	327 DHCP Discover - Transaction ID 0x1b851338

Ilustración 15: Ping PC-1 a PC-2 capturado en "eth4" del "ovs-1"

Como se ha visto en este ejemplo, Open vSwitch no nos limita a gestionar los flujos de tráfico solamente con el nivel de enlace de datos (mediante direcciones MAC, como en el escenario anterior), sino que nos permite añadir también direcciones IP y controlar así a nivel de red.

Este ejemplo es muy ilustrativo, ya que como se puede ver en el Anexo 4, somos capaces de separar el tráfico ARP gestionado con direcciones MAC, del tráfico Ethernet gestionado con direcciones IP. En este punto, uno se puede dar cuenta del gran potencial de esta tecnología, ya que, por ejemplo, en nuestro escenario no se permitiría el uso de un servidor DHCP, útil si quisiéramos configurar automáticamente las direcciones IP de los equipos. También podemos constatar que para un correcto control de los equipos es necesario un conocimiento detallado de los protocolos utilizados en la comunicación.

3.4. ESCENARIO 4 – TRÁFICO MARCADO Y DESMARCADO EN VLAN

Hasta ahora, habíamos creado un primer escenario con una sola red y ampliando este, obtuvimos el segundo, introduciendo un equipo de una red diferente. Ahora podemos dar un paso más y experimentar con la comunicación con VLAN entre varios *switch*, no solamente con uno como hasta el momento. De esta forma introducimos el marcado y desmarcado en las VLAN y los enlaces *trunk* entre *switch*.

Este es un escenario interesante para analizar, ya que en la actualidad se está adoptando el uso de VLAN cada vez con más intensidad. De este modo, mezclaremos tráfico de distintas redes virtuales, por los mismos enlaces, para comprobar cómo se desenvuelven nuestros nodos “ovs”.

Para probar el marcado (*tag*) y desmarcado (*untag*) de tráfico en las redes virtuales, ampliamos el escenario anterior. En éste mantenemos nuestro anterior escenario en la parte izquierda y añadimos a la derecha tres equipos nuevos. Un PC de la red 192.168.1.0/24, uno de la red 192.168.2.0/24 conectados entre ellos a un nuevo *switch*. De esta forma, tendremos equipos de ambas redes a cada lado de los *switch.*

Para que nuestros *switch* puedan identificar el tipo de tráfico que tienen que reenviar, hay que definir correctamente las interfaces correspondientes. Por tanto, definiremos las interfaces a las que se conectan los “Ostinato” como interfaces de acceso y las que conectan a los *switch* como *trunk* para que así puedan transportar tráfico de las dos redes.

Tras la descripción del escenario, en la siguiente Ilustración mostramos el aspecto que tendrá visualmente:

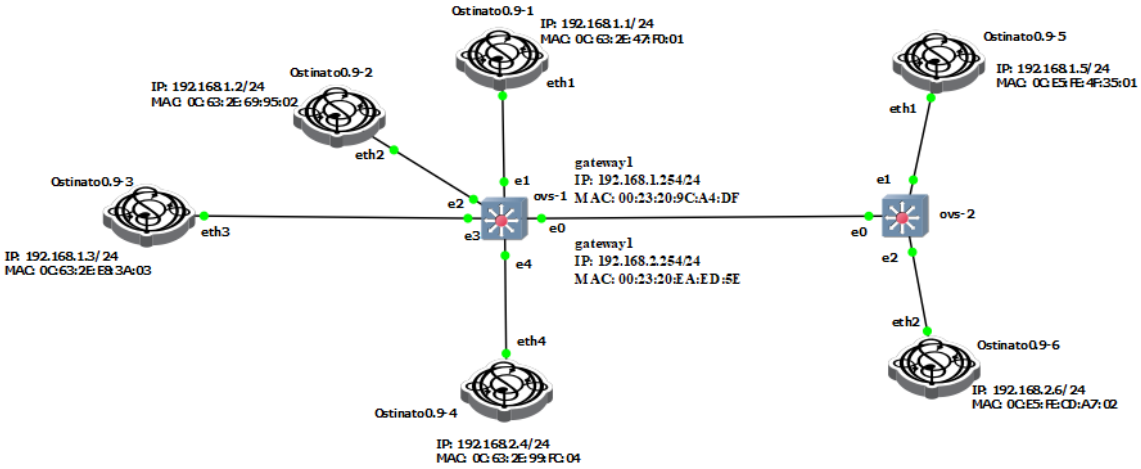


Ilustración 16: Escenario 4 - Tráfico marcado y desmarcado en VLAN

Para la configuración de este escenario, a pesar de ser una extensión del anterior, se han hecho ciertas modificaciones en el direccionamiento IP de los equipos "Ostinato". Aunque se han mantenido las direcciones, algunas de ellas ahora están configuradas en interfaces virtuales. En concreto, los PC-1 y PC-5 están configurados en las interfaces eth1.100, los PC-4 y PC-6, están en la eth4.200 y eth6.200 respectivamente y, por último, los PC-2 y PC-3, se mantienen como hasta ahora. Se puede consulta más información sobre como configurar interfaces virtuales en "Ostinato" en el Anexo 1.

La configuración de todo el escenario está descrita con detalle en el Anexo 5. En ella se pueden comprobar la creación de interfaces virtuales para marcar el tráfico en los equipos "Ostinato", la configuración de sus direcciones y también, los flujos introducidos en ambos *switch,* para ejercer de controlador e indicarles de qué modo y por dónde deben reenviar el tráfico que reciben.

Tras las pruebas realizadas en este escenario, se detectan carencias en el funcionamiento del softwre *Open vSwitch* en esta situación concreta y con la versión instalada en estas máquinas (1.2.2). No hemos obtenido los resultados esperados, ya que, aunque los PC envían tráfico marcado o desmarcado, según lo hemos configurado, los *switch* sin embargo no lo hacen correctamente. Destacamos que el problema viene dado a la hora de marcar y desmarcar el tráfico, porque el reenvío por los puertos de entrada y salida indicados, sí que funciona correctamente.

Por lo tanto, podemos concluir que no será una buena idea, adentrarse en configuraciones relativas a VLAN, con características similares en dispositivos como estos.

Sin embargo, posteriormente se ha replicado en un escenario alternativo, a muy pequeña escala (con 2 PC y 1 *switch*) la problemática del marcado y desmarcado de VLAN y se han conseguido resultados positivos con la versión 2.4.0 de Open vSwitch. Consideramos que las versiones más actuales han solventado este inconveniente, aunque no se haya profundizado con exhaustivas pruebas sobre esta versión.

3.5. ESCENARIO 5 – MÚLTIPLES CAMINOS

En este último escenario, que puede ser considerado propiamente como SD-WAN, vamos a trabajar a nivel de transporte, analizando la respuesta de los *switch* al recibir tráfico de distintos protocolos y tener que tratarlos de manera diferente. Se considera de gran utilidad el poder separar tráfico según estas características, debido a que puede darse el caso de necesitar priorizar un tráfico sobre otro si se disponen de varias rutas, haciendo que determinados tipos de tráfico se envíen por unos enlaces u otros.

Tras haber controlado el tráfico a nivel de enlace y a nivel de red en los escenarios anteriores, con este último lo realizaremos a nivel de transporte. Cubriendo así las principales problemáticas que se nos planteaban en los objetivos de este Trabajo Fin de Grado.

Para realizar las pruebas pertinentes, es necesaria la creación de un escenario, con varios *switch,* disponiendo así de variedad de rutas para nuestro tráfico. Por ser consistentes con nuestro Trabajo, hemos decidido continuar con el escenario anterior y le hemos añadido un *switch* más, conectando así todos nuestros equipos de encaminamiento. De esta forma tendremos más de un camino para poder separar el tráfico.

En la siguiente Ilustración, mostramos el aspecto de nuestro último escenario:

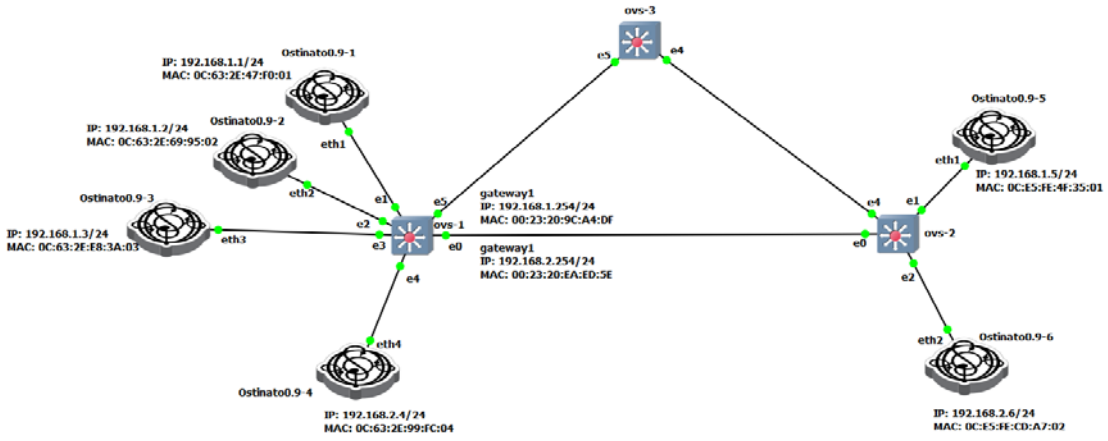


Ilustración 17: Escenario 5 - Múltiples caminos

Ciertamente este escenario podría ser configurado para actuar en cualquiera de los niveles anteriores, pero tras haberlos presentado, se cree que a nivel de transporte es en el que más partido se le puede sacar. Este escenario servirá como prueba para comprobar el funcionamiento de este tipo de flujos de control. Recordamos que todos los niveles pueden convivir en un mismo escenario mientras las órdenes no sean contradictorias.

A la hora de configurar este escenario, se ha buscado la practicidad para dar respuesta a la problemática planteada. Por tanto, se han elegido los tipos de tráfico UDP y TCP. Se configurarán los distintos *switch* para reeenviar el tráfico UDP a través del "ovs-3" y el tráfico TCP, mediante el enlace "ovs-1" – "ovs-2".

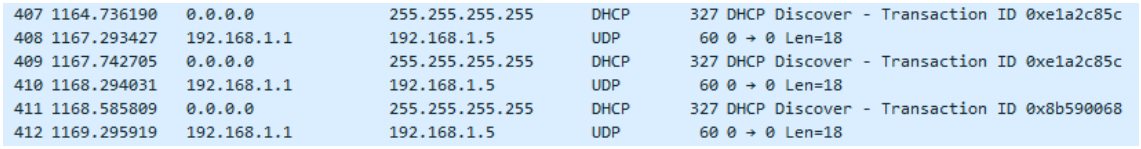
El detalle de los flujos y configuraciones de este escenario se pueden encontrar en el Anexo 6.

En este mismo escenario, se pueden mezclar y probar todas las configuraciones anteriores, teniendo en cuenta los nuevos puertos y direcciones. Esto no se considera necesario, ya que se intenta obtener un resultado concreto.

Tras varias pruebas de configuración, se podrían mostrar numerosas comunicaciones entre los equipos de este escenario. Para visualizar la correcta división del tráfico, creemos conveniente plasmar una prueba concreta en la que se vea claramente el resultado a mostrar. Para ello, lanzaremos tráfico desde “Ostinato0.9-1” a “Ostinato0.9-5”, estando cada uno en un lado del escenario y comprobamos como se comportan los diferentes flujos. Capturamos en las interfaces “eth0” y “eth5” del router “ovs-1”, de este modo podremos observar que el envío de los distintos tipos de tráfico, se realiza siguiendo rutas distintas. En primer lugar enviaremos tráfico UDP y posteriormente TCP.

Para generar el tráfico requerido, hemos utilizado la herramienta Ostinato, por ello, en las capturas siguientes, sólo observamos el tráfico enviado.

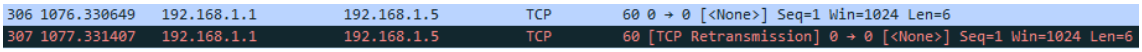
Por el enlace “ovs-1” – “ovs-3”, observamos tanto los paquetes UDP generados, como los DHCP (que también son UDP en si mismos).



407	1164.736190	0.0.0.0	255.255.255.255	DHCP	327 DHCP Discover - Transaction ID 0xe1a2c85c
408	1167.293427	192.168.1.1	192.168.1.5	UDP	60 0 → 0 Len=18
409	1167.742705	0.0.0.0	255.255.255.255	DHCP	327 DHCP Discover - Transaction ID 0xe1a2c85c
410	1168.294031	192.168.1.1	192.168.1.5	UDP	60 0 → 0 Len=18
411	1168.585809	0.0.0.0	255.255.255.255	DHCP	327 DHCP Discover - Transaction ID 0x8b590068
412	1169.295919	192.168.1.1	192.168.1.5	UDP	60 0 → 0 Len=18

Ilustración 18: Ping PC-1 a PC-5 capturado en "eth5" del "ovs-1"

Sin embargo, por el enlace “ovs-1” – “ovs-2”, observamos solamente los paquetes TCP.



306	1076.330649	192.168.1.1	192.168.1.5	TCP	60 0 → 0 [<None>] Seq=1 Win=1024 Len=6
307	1077.331407	192.168.1.1	192.168.1.5	TCP	60 [TCP Retransmission] 0 → 0 [<None>] Seq=1 Win=1024 Len=6

Ilustración 19: Ping PC-1 a PC-5 capturado en "eth0" del "ovs-1"

Mediante estas capturas, nuestra intención era mostrar cómo el funcionamiento de las órdenes de los flujos con protocolos eran funcionales. Combinando estas, con las anteriores, podemos controlar el tráfico según nuestras necesidades. Notar que las capturas solamente son un ejemplo de un caso aislado, con retrasmisiones constantes.

Por tanto, queda analizado en este último escenario el control de los flujos a nivel de transporte. Siendo éste operativo y permitiendo el encaminamiento de los diferentes tipos de tráfico, por diferentes rutas.

3.6. RESUMEN DE RESULTADOS

Tras el montaje y análisis de todos los escenarios, se procede a resumir y agrupar los resultados.

Con el Escenario 1, se ha conseguido conectar a Internet el escenario virtualizado, lo que nos proporciona libertad para instalar los software necesarios para la realización de este Trabajo Fin de Grado.

En el Escenario 2, inicialmente se ha mostrado una visión global de los equipos finales "Ostinato", conmutadores "ovs", controlador "Ubuntu" y la salida de todos estos a Internet. Posteriormente, para mejorar la percepción visual, se hansimplificado las Ilustraciones en todos los demás escenarios, aunque se hayan seguido usando para las pruebas de control y configuración. Adicionalmente, se ha analizado el tráfico a nivel de enlace, para encaminarlo a través de las direcciones MAC.

Destacamos, que desde este Escenario, en el que se ha incorporado la red de control con el equipo "Ubuntu", se permite la conexión de este a los nodos *switch* mediante "ssh". Esta herramienta nos permite conectarnos remotamente a los equipos *switch* que queremos controlar y ejecutar los comandos deseados con la creación de un *script*. Comprobando, a su vez que podemos modificar las configuraciones en tiempo real y que los flujos son reencaminados casi al instante. Los comandos necesarios están detallados en los Anexos.

Después, en el Escenario 3 se han introducido las funciones de router al *switch* convirtiendolo así en un *multilayer switch*. Esto nos ha permitido controlar el tráfico a nivel de red, mediante las direcciones IP.

Analizando el Escenario 4, se han detectado defectos en la consistencia del software utilizado. Tras varias pruebas, se ha concluido que el funcionamiento de las VLAN en esta versión de Open vSwitch (1.2.2) no es el correcto y que es necesario actualizar la versión a la 2.4.0.

Por último, en el Escenario 5, se ha analizado la posibilidad de disponer de varios caminos para el tráfico generado. De esta manera se ha separado, de acuerdo a los diferentes protoclos utilizados, para analizar las capacidades a nivel de transporte.

En general, se concluye que es una tecnología funcional y realmente útil a la hora de controlar el tráfico mediante el correcto encaminamiento de sus flujos.

4. CONCLUSIONES Y LÍNEAS FUTURAS

4.1. CONCLUSIONES

A lo largo de este Trabajo Fin de Grado, se ha realizado un estudio teórico de los sistemas SD-WAN, para posteriormente, con la información recogida, analizar las posibilidades de aplicación en entornos de red empresariales.

Como se ha detallado en la memoria, la elección de un entorno virtualizado no es casualidad. Esto nos ha permitido la realización de pruebas controladas para la posterior captura y procesamiento de datos. En cada uno de los escenarios montados, se ha proporcionado comunicación entre los generadores de tráfico, mediante el uso de órdenes concretas para cada flujo de datos en los equipos de red. Primero comprobando el correcto funcionamiento de las órdenes directamente en los *switch* y posteriormente enviándolas desde un controlador.

Durante la memoria, se han descrito de manera teórica las herramientas a utilizar y los escenarios montados, con sus respectivas pruebas. Posteriormente para ampliar esta información y documentar el trabajo realizado, en los Anexos se han detallado las configuraciones de cada uno de los equipos en los diferentes escenarios. Esta se considera una parte importante de este Trabajo, ya que puede servir como guía de configuración para introducirse en las redes definidas por software.

Destacamos la importancia del aprendizaje del uso de todas las herramientas descritas en la memoria. El conocimiento de estas, dada su flexibilidad, podrá ser muy útil, tanto para la continuación de este proyecto, como para futuros trabajos relacionados con todo tipo de redes.

Durante el Trabajo, se ha aplicado una metodología de pruebas, esto es fundamental para poder comprobar el correcto funcionamiento de las configuraciones y detectar fallos. Mediante el análisis de estas pruebas se han podido encontrar dificultades por ejemplo en el uso y configuración de VLAN en el software *Open vSwitch*. Este aspecto es de vital importancia, debido a que, en un entorno real, puede resultar mucho más interesante encontrar los fallos y limitaciones de un sistema, que limitarnos a comprobar que todo funciona correctamente.

Tras la realización de esta memoria, obtenemos un proyecto escalable, con líneas de futuro abiertas y de utilidad en el entorno empresarial. Este trabajo podrá servir de precedente para futuros acercamientos a la tecnología SD-WAN por parte de EFOR o incluso de base para poder escoger una de las tecnologías propietarias existentes. Con él, podemos concluir que el abanico de posibilidades que ofrece esta tecnología es muy amplio. Es una tecnología muy versátil, por lo que es susceptible de poder ser aplicada en numerosos entornos reales diferentes, desde pequeñas y medianas empresas, hasta grandes multinacionales, avanzando así hacia unas comunicaciones globales más inteligentes.

4.2. LÍNEAS FUTURAS

Este Trabajo Fin de Grado es un primer acercamiento de EFOR a la tecnología SD-WAN. Por tanto, se abren numerosas vías para poder continuar analizando diferentes opciones en el futuro. Queremos destacar algunas de las más útiles, desde nuestro punto de vista:

- Desarrollo del equipo controlador. Para dotarlo de mayor autonomía y poder de decisión, ya que, hasta ahora, éramos nosotros los que ejercíamos este papel. Esto requerirá una mejor comunicación entre los equipos y el controlador.

  En este punto, destacaría el poder del *machine learning* con el que nuestro controlador, basado en una correcta monitorización de la red, podría aprender de manera autónoma y así adecuar aún más cada respuesta a las necesidades de los usuarios.

- Implementación de un sistema de *backup.* En un escenario de pruebas genérico, podría ser una buena idea experimentar con los diferentes sistemas de *backup* para proteger al escenario completo de posibles caídas del controlador y así evitar fallos.

- Continuación del estudio de un protocolo general. Por ejemplo, OpenFlow con el que se obtendrán características nuevas en el control del tráfico, que hasta ahora no hemos podido desarrollar.

- Experimentar con protocolos propietarios. Otra posible continuación podrían ser las pruebas con protocolos propietarios como los de Cisco, Riverbed o Fortinet, los cuales requieren una inversión, pero son soluciones profesionales y pueden aportar mucho valor en el entorno empresarial.

BIBLIOGRAFÍA

[1] https://www.integratecnologia.es
[2] https://www.efor.es
[3] IETF 7149: "Software-Defined Networking: A Perspective from within a Service Provider Environment". https://tools.ietf.org/html/rfc7149, marzo de 2014.
[4] https://www.lainnovacionnecesaria.com/sd-wan-el-presente-de-las-comunicaciones/
[5] Turban, E; King, D; Lee, J; Viehland, D (2008). «Chapter 19: Building E-Commerce Applications and Infrastructure». Electronic Commerce A Managerial Perspective (5th edición). Prentice-Hall. p. 27.
[6] Welsh, "RedNectar" Chris (2013). GNS3 Network Simulation Guide. Packt Publishing Ltd. ISBN 9781782160816.]
[7] Neumann, Jason C. (2015). The Book of GNS3: Build Virtual Network Labs Using Cisco, Juniper, and More. No Starch Press. ISBN 9781593276959.
[8] https://wiki.qemu.org/
[9] https://es.wikipedia.org/wiki/QEMU
[10] https://www.vmware.com/es/products.html
[11] http://wiki.tinycorelinux.net/
[12] https://es.wikipedia.org/wiki/Ubuntu
[13] https://gns3.com/marketplace/appliance/ubuntu-with-gui
[14] https://www.wireshark.org/docs/wsug_html_chunked/
[15] http://www.openvswitch.org/en/latest/
[16] https://www.ssh.com/ssh/

Respecto a las configuraciones de los equipos, otras páginas web útiles han sido:
[a] https://docs.pica8.com/pages/viewpage.action?pageId=3083175
[b] http://therandomsecurityguy.com/openvswitch-cheat-sheet/
[c] http://www.openvswitch.org/support/dist-docs/
[d] http://luisarizmendi.blogspot.com/2014
[e] http://luisarizmendi.blogspot.com/2014/07/kvm-y-open-vswitch-ovs-componentes.html

Todas las páginas web han sido visitadas por última vez en 20 de septiembre de 2019.

ANEXO 1 – CONFIGURACIÓN DIRECCIONAMIENTO IP

OSTINATO 0.9 – Tiny Core Linux

En los nodos "Ostinato" tenemos dos métodos principales para configurar el direccionamiento IP:

- El primero, dado que estos equipos disponen de interfaz gráfica, se trata de abrir el panel de control y en la pestaña *Network* e introducir los valores correspondientes a la interfaz que queremos configurar, dirección IP, máscara, etc.

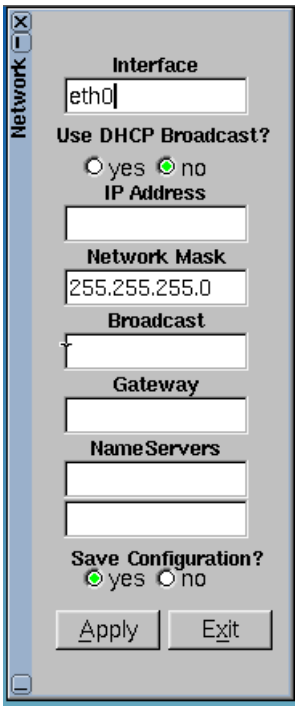


Ilustración 20: Configuración de direcciones mediante Interfaz gráfica

- El segundo método es mediante el terminal. Podemos abrir un terminal en el equipo, **como en cualquier otro equipo Linux** y mediante los siguientes comandos configurar las mismas direcciones descritas anteriormente. También podemos activarlas y desactivarlas e incluso borrar las direcciones o los interfaces (si son virtuales).

  sudo ifconfig <interfaz> <dirección IP> netmask <máscara> broadcast <dirección *broadcast*> up

Para borrar la dirección IP:

sudo ip addr del <dirección IP></máscara> dev <interfaz>

Para añadir una interfaz virtual:

sudo vconfig add <interfaz> <vlan>
ifconfig <interfaz.vlan>

Ahora añadimos las direcciones correspondientes:

sudo ifconfig <interfaz> <dirección IP> netmask <máscara> broadcast <dirección *broadcast*> up

Para borrar la interfaz (si es virtual):

ifconfig <interfaz> down
sudo vconfig rem <interfaz>

Para salvar la configuración de direcciones realizada en cada equipo "Ostinato", se recomienda apagarlos de manera correcta, sin forzar su cierre, mediante "Exit" y "Shutdown".

OPEN VSWITCH – Micro Core Linux

En nuestro caso, el nodo "ovs" no tiene interfaz gráfica, por tanto, tendríamos que usar el segundo método explicado para los nodos "Ostinato", ya que se configura mediante el terminal y también son equipos Linux.

Hay que destacar que, a estos equipos, además de poder configurar direcciones en sus interfaces externas, si están siendo utilizadas por algún *brigde* solamente podemos definirles direcciones IP cuando se trata de interfaces definidas como internas. Esto es útil, por ejemplo, cuando queremos que el *switch* haga las funciones de *router* como en el Escenario 3.

Para definir una interfaz como interna y así poderle añadir direcciones tendríamos que hacer lo siguiente:

sudo ovs-vsctl add-port <*bridge*> <interfaz> -- set interface <interfaz> type=internal

Después podríamos definir la dirección IP:

sudo ifconfig <interfaz> <dirección IP> netmask <máscara> up

UBUNTU

En Ubuntu, como se trata también de un equipo Linux, podremos seguir usando el método general, descrito en el segundo caso de Ostinato. Además, otro método adicional será acceder al fichero *interfaces*, cambiar las direcciones de todos los interfaces deseados y posteriormente reiniciar las interfaces de red. Esto se puede hacer con:

nano /etc/network/interfaces
service networking restart

ANEXO 2 – CONEXIÓN MEDIANTE SSH

Para permitir la conexión mediante SSH, tanto en el equipo Ubuntu que hará de cliente, como los Micro Core Linux que hará de servidor deben tener instalados "ssh". Los pasos para ejecutar el servidor en los equipos Micro Core Linux son:

Comprobamos que está instalado:

```
tce-load -wi openssh
```

Creamos copias de los archivos necesarios para ejecutarlo:

```
cd /usr/local/etc/ssh/
sudo cp ssh_config.example ssh_config
sudo cp sshd_config.example sshd_config
```

Lanzamos el servidor ssh:

```
sudo /usr/local/etc/init.d/openssh start
```

Cambiamos la contraseña del usuario al que nos querremos conectar. En este caso tc, pero podría ser el de su.

```
passwd
```

En caso de querer cambiar la contraseña del root, para conectarnos a él:

```
sudo su passwd
```

Chequeamos que las direcciones IP sean las correctas para poder conectarnos;

```
ifconfig
```

Verificamos que el servicio "ssh" está activo y escuchando en el puerto 22:

```
sudo netstat -anp | grep 22
```

Por otro lado, desde el controlador Ubuntu con el que nos queremos conectar al *switch* (equipo Micro Core Linux que acabamos de configurar y en que hemos lanzado el servidor), introduciremos el comando y se nos requerirá la contraseña definida anteriormente:

```
ssh tc@<dirección ip>
```

Una vez conectados, podremos tanto ejecutar directamente los comandos como si configuraramos el *switch*, como realizar un "scp" para copiar en el destino un documento con todos los comandos que deseamos introducir y ejecutarlos. Más información de "scp" en: https://www.garron.me/es/articulos/scp.html

Ahora tendremos el control de los Micro Core Linux (*switch*), desde el equipo Ubuntu y podremos configurar flujos para controlar el tráfico entre otras cosas. Como ejemplo, podemos introducir flujos desde un fichero con la orden:

```
sudo ovs-ofctl replace-flows SWITCH FILE
```

Para una optimización en el funcionamiento de este recurso, esta configuración se puede hacer permanente. Esta configuración se puede ver a fondo en la página web: https://iotbytes.wordpress.com/configure-ssh-server-on-microcore-tiny-linux/

ANEXO 3 – *LEARNING SWITCH*

Para la configuración del Escenario 2, las direcciones IP de los equipos "Ostinato" se configurarán de la siguiente manera:

En el "Ostinato0.9-1":

```
sudo ifconfig eth1 192.168.1.1 netmask 255.255.255.0 broadcast 192.168.1.254
up
```

En el "Ostinato0.9-2":

```
sudo ifconfig eth2 192.168.1.2 netmask 255.255.255.0 broadcast 192.168.1.254
up
```

En el "Ostinato0.9-3":

```
sudo ifconfig eth3 192.168.1.3 netmask 255.255.255.0 broadcast 192.168.1.254
up
```

Después, la configuración en el nodo "ovs" para dotarlo de funcionalidad de *learning switch* será la siguiente:

Se crea un *bridge* y se añaden los interfaces correspondientes:

```
sudo ovs-vsctl add-br br0
sudo ovs-vsctl add-port br0 eth1
sudo ovs-vsctl add-port br0 eth2
sudo ovs-vsctl add-port br0 eth3
```

Para ver los *bridges* e interfaces que tenemos creados en nuestro "ovs":

```
sudo ovs-vsctl show
```

Los puertos que utilizaremos a la hora de añadir los flujos en el Open vSwitch, no tienen por qué coincidir con el numero de la interfaz ni a los puertos del switch a los que está conectada. Por tanto, para ver estos puertos hay que utilizar la siguiente orden:

```
sudo ovs-vsctl -- --columns=name,ofport list Interface
```

Y si se quiere de un solo interfaz en concreto:

```
sudo ovs-vsctl get Interface <interfaz> ofport
```

Después añadimos los flujos correspondientes. En este caso, como ya hemos explicado anteriormente, definimos el puerto de entrada, el de salida y estará condicionado a las direcciones MAC.

```
sudo ovs-ofctl add-flow <bridge> in_port=<puerto>,dl_dst=<dirección
MAC>,actions=output: =<puerto/s>
```

```
sudo ovs-ofctl add-flow br0 in_port=1,dl_dst=ff:ff:ff:ff:ff:ff,actions=output:2,3
sudo ovs-ofctl add-flow br0
in_port=1,dl_dst=0C:63:2E:69:95:02,actions=output:3
```

```
sudo ovs-ofctl add-flow br0
in_port=1,dl_dst=0C:63:2E:E8:3A:03,actions=output:2
sudo ovs-ofctl add-flow br0 in_port=3,dl_dst=ff:ff:ff:ff:ff:ff,actions=output:1,2
sudo ovs-ofctl add-flow br0
in_port=3,dl_dst=0C:63:2E:47:F0:01,actions=output:1
sudo ovs-ofctl add-flow br0
in_port=3,dl_dst=0C:63:2E:E8:3A:03,actions=output:2
sudo ovs-ofctl add-flow br0 in_port=2,dl_dst=ff:ff:ff:ff:ff:ff,actions=output:1,3
sudo ovs-ofctl add-flow br0
in_port=2,dl_dst=0C:63:2E:47:F0:01,actions=output:1
sudo ovs-ofctl add-flow br0
in_port=2,dl_dst=0C:63:2E:69:95:02,actions=output:3
```

Y finalmente para guardar esta configuración ejecutamos el siguiente comando:

```
filetool.sh -b
```

NOTA: Tanto en esta, como en todas las configuraciones, hay que tener presente que son ejemplos. Por tanto, si se quieren replicar hay que tener muy en cuenta los puertos de entrada y salida, además de las direcciones MAC de las máquinas en cada caso.

ANEXO 4 – *SWITCH* CON FUNCIONES DE ROUTER

En el Escenario 3, la configuración de los equipos "Ostinato" será igual que la anterior y se le añadirá un nuevo PC en otra nueva red:

En el "Ostinato0.9-1":

```
sudo ifconfig eth1 192.168.1.1 netmask 255.255.255.0 broadcast 192.168.1.254
up
```

En el "Ostinato0.9-2":

```
sudo ifconfig eth2 192.168.1.2 netmask 255.255.255.0 broadcast 192.168.1.254
up
```

En el "Ostinato0.9-3":

```
sudo ifconfig eth3 192.168.1.3 netmask 255.255.255.0 broadcast 192.168.1.254
up
```

En el "Ostinato0.9-4":

```
sudo ifconfig eth4 192.168.2.4 netmask 255.255.255.0 broadcast 192.168.2.254
up
```

Posteriormente, para configurar el nodo "ovs" y dotar al *switch* con funciones de router se realiza la siguiente configuración. Para su mejor comprensión, aunque sea una extensión del Escenario 2, proponemos la configuración total, como si se tratase de un escenario totalmente nuevo:

Creamos el *bridge* y le añadimos las interfaces correspondientes:

```
sudo ovs-vsctl add-br br0
sudo ovs-vsctl add-port br0 eth1
sudo ovs-vsctl add-port br0 eth2
sudo ovs-vsctl add-port br0 eth3
sudo ovs-vsctl add-port br0 eth4
```

Siempre que se quieran conectar dos o más redes IP, es necesario un *router*. Como en este caso lo hemos creado dentro de nuestro *switch* necesitamos crear interfaces de tipo interno, para poder asignarles direcciones IP y que así sean accesibles a la red.

```
sudo ovs-vsctl add-port br0 gateway1 -- set interface gateway1 type=internal
sudo ovs-vsctl add-port br0 gateway2 -- set interface gateway2 type=internal
sudo ovs-vsctl show

sudo ifconfig gateway1 192.168.1.254 netmask 255.255.255.0 up
sudo ifconfig gateway2 192.168.2.254 netmask 255.255.255.0 up
ifconfig
```

Aunque en este escenario podríamos continuar utilizando solamente las direcciones MAC para definir los flujos y gestionar el tráfico, queremos explorar más posibilidades. Por ello, pasamos a trabajar con el nivel de red y comenzamos a utilizar también direcciones IP. Es una manera de hacer más flexible esta tecnología y seguir explorando más de las posibilidades que nos brinda.

Activamos el *forwarding,* con el comando:

```
sudo sysctl -w net.ipv4.ip_forward=1
```

Ahora tendremos que añadir los flujos correspondientes, que, para este ejemplo, con los correspondientes puertos en el Open vSwitch son:

```
sudo ovs-vsctl -- --columns=name,ofport list Interface
```

Flujos relativos a la entrada de tráfico por la interfaz 1 (puerto 1), del tipo ARP (0x0806) y Ethernet (0x0800):

```
sudo ovs-ofctl add-flow br0
in_port=1,dl_type=0x0806,dl_dst=ff:ff:ff:ff:ff:ff,actions=output:4,3,5
sudo ovs-ofctl add-flow br0
in_port=1,dl_type=0x0806,dl_dst=0C:63:2E:69:95:02,actions=output:4
sudo ovs-ofctl add-flow br0
in_port=1,dl_type=0x0806,dl_dst=0C:63:2E:E8:3A:03,actions=output:3
sudo ovs-ofctl add-flow br0
in_port=1,dl_type=0x0806,dl_dst=00:23:20:9C:A4:DF,actions=output:5

sudo ovs-ofctl add-flow br0
in_port=1,dl_type=0x0800,nw_dst=192.168.1.2,actions=output:4
sudo ovs-ofctl add-flow br0
in_port=1,dl_type=0x0800,nw_dst=192.168.1.3,actions=output:3
sudo ovs-ofctl add-flow br0
in_port=1,dl_type=0x0800,nw_dst=192.168.1.254,actions=output:5
sudo ovs-ofctl add-flow br0
in_port=1,dl_type=0x0800,nw_dst=192.168.2.0/24,actions=output:5
```

Flujos relativos a la entrada de tráfico por la interfaz 2 (puerto 4), del tipo ARP (0x0806) y Ethernet (0x0800):

```
sudo ovs-ofctl add-flow br0
in_port=4,dl_type=0x0806,dl_dst=ff:ff:ff:ff:ff:ff,actions=output:1,3,5
sudo ovs-ofctl add-flow br0
in_port=4,dl_type=0x0806,dl_dst=0C:63:2E:47:F0:01,actions=output:1
sudo ovs-ofctl add-flow br0
in_port=4,dl_type=0x0806,dl_dst=0C:63:2E:E8:3A:03,actions=output:3
sudo ovs-ofctl add-flow br0
in_port=4,dl_type=0x0806,dl_dst=00:23:20:9C:A4:DF,actions=output:5

sudo ovs-ofctl add-flow br0
in_port=4,dl_type=0x0800,nw_dst=192.168.1.1,actions=output:1
sudo ovs-ofctl add-flow br0
in_port=4,dl_type=0x0800,nw_dst=192.168.1.3,actions=output:3
```

```
sudo ovs-ofctl add-flow br0
in_port=4,dl_type=0x0800,nw_dst=192.168.1.254,actions=output:5
sudo ovs-ofctl add-flow br0
in_port=4,dl_type=0x0800,nw_dst=192.168.2.0/24,actions=output:5
```

Flujos relativos a la entrada de tráfico por la interfaz 3 (puerto 3), del tipo ARP (0x0806) y Ethernet (0x0800):

```
sudo ovs-ofctl add-flow br0
in_port=3,dl_type=0x0806,dl_dst=ff:ff:ff:ff:ff:ff,actions=output:1,4,5
sudo ovs-ofctl add-flow br0
in_port=3,dl_type=0x0806,dl_dst=0C:63:2E:47:F0:01,actions=output:1
sudo ovs-ofctl add-flow br0
in_port=3,dl_type=0x0806,dl_dst=0C:63:2E:69:95:02,actions=output:4
sudo ovs-ofctl add-flow br0
in_port=3,dl_type=0x0806,dl_dst=00:23:20:9C:A4:DF,actions=output:5

sudo ovs-ofctl add-flow br0
in_port=3,dl_type=0x0800,nw_dst=192.168.1.1,actions=output:1
sudo ovs-ofctl add-flow br0
in_port=3,dl_type=0x0800,nw_dst=192.168.1.2,actions=output:4
sudo ovs-ofctl add-flow br0
in_port=3,dl_type=0x0800,nw_dst=192.168.1.254,actions=output:5
sudo ovs-ofctl add-flow br0
in_port=3,dl_type=0x0800,nw_dst=192.168.2.0/24,actions=output:5
```

Flujos relativos a la entrada de tráfico por la interfaz interna “gateway1” (puerto 5), del tipo ARP (0x0806) y Ethernet (0x0800):

```
sudo ovs-ofctl add-flow br0
in_port=5,dl_type=0x0806,dl_dst=ff:ff:ff:ff:ff:ff,actions=output:1,4,3
sudo ovs-ofctl add-flow br0
in_port=5,dl_type=0x0806,dl_dst=0C:63:2E:47:F0:01,actions=output:1
sudo ovs-ofctl add-flow br0
in_port=5,dl_type=0x0806,dl_dst=0C:63:2E:69:95:02,actions=output:4
sudo ovs-ofctl add-flow br0
in_port=5,dl_type=0x0806,dl_dst=0C:63:2E:E8:3A:03,actions=output:3

sudo ovs-ofctl add-flow br0
in_port=5,dl_type=0x0800,nw_dst=192.168.1.1,actions=output:1
sudo ovs-ofctl add-flow br0
in_port=5,dl_type=0x0800,nw_dst=192.168.1.2,actions=output:4
sudo ovs-ofctl add-flow br0
in_port=5,dl_type=0x0800,nw_dst=192.168.1.3,actions=output:3
```

Flujos relativos a la entrada de tráfico por la interfaz interna “gateway2” (puerto 6), del tipo ARP (0x0806) y Ethernet (0x0800):

```
sudo ovs-ofctl add-flow br0
in_port=6,dl_type=0x0806,dl_dst=ff:ff:ff:ff:ff:ff,actions=output:2
sudo ovs-ofctl add-flow br0
in_port=6,dl_type=0x0806,dl_dst=00:23:20:EA:ED:5E,actions=output:2
```

```
sudo ovs-ofctl add-flow br0
in_port=6,dl_type=0x0800,nw_dst=192.168.2.254,actions=output:2
sudo ovs-ofctl add-flow br0
in_port=6,dl_type=0x0800,nw_dst=192.168.1.0/24,actions=output:2
```

Flujos relativos a la entrada de tráfico por la interfaz 4 (puerto 2), del tipo ARP (0x0806) y Ethernet (0x0800):

```
sudo ovs-ofctl add-flow br0
in_port=2,dl_type=0x0806,dl_dst=ff:ff:ff:ff:ff:ff,actions=output:6
sudo ovs-ofctl add-flow br0
in_port=2,dl_type=0x0806,dl_dst=0C:63:2E:99:FC:04,actions=output:6

sudo ovs-ofctl add-flow br0
in_port=2,dl_type=0x0800,nw_dst=192.168.2.4,actions=output:6
```

Para hacer persistentes estos cambios:

```
filetool.sh -b
```

NOTA: Tanto en esta, como en todas las configuraciones, hay que tener presente que son ejemplos. Por tanto, si se quieren replicar hay que tener muy en cuenta los puertos de entrada y salida, además de las direcciones MAC de las máquinas en cada caso.

ANEXO 5 – TRÁFICO MARCADO Y DESMARCADO EN VLAN

En el Escenario 4, la configuración de los equipos "Ostinato" variará ligeramente de las anteriores, por incluir interfaces virtuales. Recordar borrar las anteriores direcciones en caso de ser necesario.

En el "Ostinato0.9-1":

```
sudo ip addr del 192.168.1.1/24 dev eth1
sudo vconfig add 1 100
sudo ifconfig eth1.100 192.168.1.1 netmask 255.255.255.0 broadcast 192.168.1.254 up
```

En el "Ostinato0.9-2":

```
sudo ifconfig eth2 192.168.1.2 netmask 255.255.255.0 broadcast 192.168.1.254 up
```

En el "Ostinato0.9-3":

```
sudo ifconfig eth3 192.168.1.3 netmask 255.255.255.0 broadcast 192.168.1.254 up
```

En el "Ostinato0.9-4":

```
sudo ip addr del 192.168.2.4/24 dev eth4
sudo vconfig add 4 200
sudo ifconfig eth4.200 192.168.2.4 netmask 255.255.255.0 broadcast 192.168.2.254 up
```

En el "Ostinato0.9-5":

```
sudo vconfig add 1 100
sudo ifconfig eth1.100 192.168.1.5 netmask 255.255.255.0 broadcast 192.168.1.254 up
```

En el "Ostinato0.9-6":

```
sudo vconfig add 2 200
sudo ifconfig eth2.200 192.168.2.6 netmask 255.255.255.0 broadcast 192.168.2.254 up
```

Posteriormente, para configurar el nodo "ovs-1" y dotar al *switch* con funciones de router y poder comunicarse con el otro *switch*, se realiza la siguiente configuración.

Creamos el *bridge* y le añadimos las interfaces correspondientes:

```
sudo ovs-vsctl add-br br0
sudo ovs-vsctl add-port br0 eth1 tag=100
sudo ovs-vsctl add-port br0 eth2
sudo ovs-vsctl add-port br0 eth3
sudo ovs-vsctl add-port br0 eth4 tag=200
sudo ovs-vsctl add-port br0 eth0 trunk=100,200
```

Siempre que se quieran conectar dos o más redes IP, es necesario un *router*. Como en este caso lo hemos creado dentro de nuestro *switch* necesitamos crear interfaces de tipo interno, para poder asignarles direcciones IP y que así sean accesibles a la red.

```
sudo ovs-vsctl add-port br0 gateway1 -- set interface gateway1 type=internal
sudo ovs-vsctl add-port br0 gateway2 -- set interface gateway2 type=internal
sudo ovs-vsctl show

sudo ifconfig gateway1 192.168.1.254 netmask 255.255.255.0 up
sudo ifconfig gateway2 192.168.2.254 netmask 255.255.255.0 up
ifconfig
```

Activamos el *forwarding,* con el comando:

```
sudo sysctl -w net.ipv4.ip_forward=1
```

Ahora tendremos que añadir los flujos correspondientes, que, para este ejemplo, con los correspondientes puertos en el Open vSwitch y direcciones son:

```
sudo ovs-vsctl -- --columns=name,ofport list Interface
```

Flujos relativos a la entrada de tráfico por la interfaz “eth1” (puerto 1):

```
sudo ovs-ofctl add-flow br0 in_port=1,dl_vlan=100,actions=output:5
sudo ovs-ofctl add-flow br0 in_port=1,dl_vlan=100,actions=strip_vlan,output:2
sudo ovs-ofctl add-flow br0 in_port=1,dl_vlan=100,actions=strip_vlan,output:3
sudo ovs-ofctl add-flow br0 in_port=1,dl_vlan=100,actions=output:6
```

Flujos relativos a la entrada de tráfico por la interfaz “eth2” (puerto 2):

```
sudo ovs-ofctl add-flow br0 in_port=2,actions=mod_vlan_vid:100,output:1
sudo ovs-ofctl add-flow br0 in_port=2,actions=output:3
sudo ovs-ofctl add-flow br0 in_port=2,actions= mod_vlan_vid:100,output:5
sudo ovs-ofctl add-flow br0 in_port=2,actions=output:6
```

Flujos relativos a la entrada de tráfico por la interfaz “eth3” (puerto 3):

```
sudo ovs-ofctl add-flow br0 in_port=3,actions=mod_vlan_vid:100,output:1
sudo ovs-ofctl add-flow br0 in_port=3,actions=output:2
sudo ovs-ofctl add-flow br0 in_port=3,actions=mod_vlan_vid:100,output:5
sudo ovs-ofctl add-flow br0 in_port=3,actions=output:6
```

Flujos relativos a la entrada de tráfico por la interfaz “eth4” (puerto 4):

```
sudo ovs-ofctl add-flow br0 in_port=4,dl_vlan=200,actions=mod_vlan_vid:200,output:5
sudo ovs-ofctl add-flow br0 in_port=4,dl_vlan=200,actions= strip_vlan,output:7
```

Flujos relativos a la entrada de tráfico por la interfaz “eth0” (puerto 5), *trunk*:

```
sudo ovs-ofctl add-flow br0 in_port=5,dl_vlan=100,actions=mod_vlan_vid:100,output:1
sudo ovs-ofctl add-flow br0 in_port=5,dl_vlan=100,actions=strip_vlan,output:2
sudo ovs-ofctl add-flow br0 in_port=5,dl_vlan=100,actions=strip_vlan,output:3
sudo ovs-ofctl add-flow br0 in_port=5,dl_vlan=100,actions=strip_vlan,output:6
sudo ovs-ofctl add-flow br0 in_port=5,dl_vlan=200,actions=mod_vlan_vid:200,output:4
```

sudo ovs-ofctl add-flow br0 in_port=5,dl_vlan=200,actions=strip_vlan,output:7

Flujos relativos a la entrada de tráfico por la interfaz interna "gateway1" (puerto 6):

sudo ovs-ofctl add-flow br0 in_port=6,nw_dst=192.168.2.0/24,actions=output:7

Flujos relativos a la entrada de tráfico por la interfaz interna "gateway2" (puerto 7):

sudo ovs-ofctl add-flow br0 in_port=7,nw_dst=192.168.1.0/24actions=output:6

Para hacer persistentes estos cambios:

filetool.sh -b

Por último, para configurar el nodo "ovs-2" y permitir al *switch* la comunicación de los PC conectados a él, con los del otro *switch* se realiza la siguiente configuración:

Creamos el *bridge* y le añadimos las interfaces correspondientes:

```
sudo ovs-vsctl add-br br0
sudo ovs-vsctl add-port br0 eth1 tag=100
sudo ovs-vsctl add-port br0 eth2 tag=200
sudo ovs-vsctl add-port br0 eth0 trunk=100,200
sudo ovs-vsctl show
```

Añadimos los flujos correspondientes, que, para este ejemplo, con los correspondientes puertos en el Open vSwitch y direcciones son:

sudo ovs-vsctl -- --columns=name,ofport list Interface

Flujos relativos a la entrada de tráfico por la interfaz "eth1" (puerto 1):

sudo ovs-ofctl add-flow br0 in_port=1,dl_vlan=100,actions=mod_vlan_vid:100,output:3

Flujos relativos a la entrada de tráfico por la interfaz "eth2" (puerto 2):

sudo ovs-ofctl add-flow br0 in_port=2,dl_vlan=200,actions=mod_vlan_vid:200,output:3

Flujos relativos a la entrada de tráfico por la interfaz "eth0" (puerto 3), *trunk*:

sudo ovs-ofctl add-flow br0 in_port=3,dl_vlan=100,actions=mod_vlan_vid:100,output:1

sudo ovs-ofctl add-flow br0 in_port=3,dl_vlan=200,actions=mod_vlan_vid:200,output:2

Para hacer persistentes estos cambios:

filetool.sh -b

NOTA: Tanto en esta, como en todas las configuraciones, hay que tener presente que son ejemplos. Por tanto, si se quieren replicar hay que tener muy en cuenta los puertos de entrada y salida, además de las direcciones MAC de las máquinas en cada caso.

ANEXO 6 – MÚLTIPLES CAMINOS

En el Escenario 5, la configuración de los equipos "Ostinato" se realizará en las interfaces reales. Recordar borrar las anteriores direcciones en caso de ser necesario.

En el "Ostinato0.9-1":

```
sudo ip addr del 192.168.1.1/24 dev eth1.100
sudo ifconfig eth1 192.168.1.1 netmask 255.255.255.0 broadcast 192.168.1.254 up
```

En el "Ostinato0.9-2":

```
sudo ifconfig eth2 192.168.1.2 netmask 255.255.255.0 broadcast 192.168.1.254 up
```

En el "Ostinato0.9-3":

```
sudo ifconfig eth3 192.168.1.3 netmask 255.255.255.0 broadcast 192.168.1.254 up
```

En el "Ostinato0.9-4":

```
sudo ip addr del 192.168.2.4/24 dev eth4.200
sudo ifconfig eth4 192.168.2.4 netmask 255.255.255.0 broadcast 192.168.2.254 up
```

En el "Ostinato0.9-5":

```
sudo ip addr del 192.168.1.5/24 dev eth1.100
sudo ifconfig eth1 192.168.1.5 netmask 255.255.255.0 broadcast 192.168.1.254 up
```

En el "Ostinato0.9-6":

```
sudo ip addr del 192.168.2.6/24 dev eth2.200
sudo ifconfig eth2 192.168.2.6 netmask 255.255.255.0 broadcast 192.168.2.254 up
```

Posteriormente, para configurar el nodo "ovs-1" y dotar al *switch* con funciones de router y poder comunicarse con el resto de *switch*, se realiza la siguiente configuración.

Creamos el *bridge* y le añadimos las interfaces correspondientes:

```
sudo ovs-vsctl add-br br0
sudo ovs-vsctl add-port br0 eth1
sudo ovs-vsctl add-port br0 eth2
sudo ovs-vsctl add-port br0 eth3
sudo ovs-vsctl add-port br0 eth4
sudo ovs-vsctl add-port br0 eth0
sudo ovs-vsctl add-port br0 eth5
```

Siempre que se quieran conectar dos o más redes IP, es necesario un *router*. Como en este caso lo hemos creado dentro de nuestro *switch* necesitamos crear interfaces de tipo interno, para poder asignarles direcciones IP y que así sean accesibles a la red.

```
sudo ovs-vsctl add-port br0 gateway1 -- set interface gateway1 type=internal
sudo ovs-vsctl add-port br0 gateway2 -- set interface gateway2 type=internal
sudo ovs-vsctl show

sudo ifconfig gateway1 192.168.1.254 netmask 255.255.255.0 up
sudo ifconfig gateway2 192.168.2.254 netmask 255.255.255.0 up
ifconfig
```

Activamos el *forwarding,* con el comando:

```
sudo sysctl -w net.ipv4.ip_forward=1
```

Ahora tendremos que añadir los flujos correspondientes, que, para este ejemplo, con los correspondientes puertos en el Open vSwitch y direcciones son:

```
sudo ovs-vsctl -- --columns=name,ofport list Interface
```

Flujos relativos a la entrada de tráfico por la interfaz “eth1” (puerto 1):

```
sudo ovs-ofctl add-flow br0 in_port=1,nw_dst=192.168.1.2/24,actions=output:2
sudo ovs-ofctl add-flow br0 in_port=1,nw_dst=192.168.1.3/24,actions=output:3
sudo ovs-ofctl add-flow br0 in_port=1,nw_dst=192.168.2.0/24,actions=output:6
sudo ovs-ofctl add-flow br0 in_port=1,dl_type=0x0800,nw_proto=6,nw_dst=192.168.1.5/24,actions=output:5
sudo ovs-ofctl add-flow br0 in_port=1,dl_type=0x0800,nw_proto=17,nw_dst=192.168.1.5/24,actions=output:8
```

Flujos relativos a la entrada de tráfico por la interfaz “eth2” (puerto 2):

```
sudo ovs-ofctl add-flow br0 in_port=2,nw_dst=192.168.1.1/24,actions=output:1
sudo ovs-ofctl add-flow br0 in_port=2,nw_dst=192.168.1.3/24,actions=output:3
sudo ovs-ofctl add-flow br0 in_port=2,nw_dst=192.168.2.0/24,actions=output:6
sudo ovs-ofctl add-flow br0 in_port=2,dl_type=0x0800,nw_proto=6,nw_dst=192.168.1.5/24,actions=output:5
sudo ovs-ofctl add-flow br0 in_port=2,dl_type=0x0800,nw_proto=17,nw_dst=192.168.1.5/24,actions=output:8
```

Flujos relativos a la entrada de tráfico por la interfaz “eth3” (puerto 3):

```
sudo ovs-ofctl add-flow br0 in_port=3,nw_dst=192.168.1.1/24,actions=output:1
sudo ovs-ofctl add-flow br0 in_port=3,nw_dst=192.168.1.2/24,actions=output:2
sudo ovs-ofctl add-flow br0 in_port=3,nw_dst=192.168.2.0/24,actions=output:6
sudo ovs-ofctl add-flow br0 in_port=3,dl_type=0x0800,nw_proto=6,nw_dst=192.168.1.5/24,actions=output:5
sudo ovs-ofctl add-flow br0 in_port=3,dl_type=0x0800,nw_proto=17,nw_dst=192.168.1.5/24,actions=output:8
```

Flujos relativos a la entrada de tráfico por la interfaz “eth4” (puerto 4):

```
sudo ovs-ofctl add-flow br0 in_port=4,nw_dst=192.168.1.0/24,actions=output:7
sudo ovs-ofctl add-flow br0 in_port=4,dl_type=0x0800,nw_proto=6,nw_dst=192.168.2.6/24,actions=output:5
sudo ovs-ofctl add-flow br0 in_port=4,dl_type=0x0800,nw_proto=17,nw_dst=192.168.2.6/24,actions=output:8
```

Flujos relativos a la entrada de tráfico por la interfaz “eth0” (puerto 5):

```
sudo ovs-ofctl add-flow br0 in_port=5,dl_type=0x0800,nw_proto=6,nw_dst=192.168.1.0/24,actions=output:6
sudo ovs-ofctl add-flow br0 in_port=5,dl_type=0x0800,nw_proto=6,nw_dst=192.168.2.0/24,actions=output:7
```

Flujos relativos a la entrada de tráfico por la interfaz interna “gatewat1” (puerto 6):

```
sudo ovs-ofctl add-flow br0 in_port=6, nw_dst=192.168.1.1/24,actions=output:1
sudo ovs-ofctl add-flow br0 in_port=6, nw_dst=192.168.1.2/24,actions=output:2
sudo ovs-ofctl add-flow br0 in_port=6, nw_dst=192.168.1.3/24,actions=output:3
sudo ovs-ofctl add-flow br0 in_port=6,dl_type=0x0800,nw_proto=6,nw_dst=192.168.1.5/24,actions=output:5
sudo ovs-ofctl add-flow br0 in_port=6,dl_type=0x0800,nw_proto=17,nw_dst=192.168.1.5/24,actions=output:8
sudo ovs-ofctl add-flow br0 in_port=6,nw_dst=192.168.2.0/24,actions=output:7
```

Flujos relativos a la entrada de tráfico por la interfaz interna “gatewat2” (puerto 7):

```
sudo ovs-ofctl add-flow br0 in_port=7, nw_dst=192.168.2.4/24,actions=output:4
sudo ovs-ofctl add-flow br0 in_port=7,dl_type=0x0800,nw_proto=6,nw_dst=192.168.2.6/24,actions=output:5
sudo ovs-ofctl add-flow br0 in_port=7,dl_type=0x0800,nw_proto=17,nw_dst=192.168.2.6/24,actions=output:8
sudo ovs-ofctl add-flow br0 in_port=7, nw_dst=192.168.1.0/24,actions=output:6
```

Flujos relativos a la entrada de tráfico por la interfaz “eth5” (puerto 8):

```
sudo ovs-ofctl add-flow br0 in_port=8,dl_type=0x0800,nw_proto=17,nw_dst=192.168.1.0/24,actions=output:6
sudo ovs-ofctl add-flow br0 in_port=8,dl_type=0x0800,nw_proto=17,nw_dst=192.168.2.0/24,actions=output:7
```

Para hacer persistentes estos cambios:

```
filetool.sh -b
```

Para configurar el nodo “ovs-2”:

Creamos el *bridge* y le añadimos las interfaces correspondientes:

```
sudo ovs-vsctl add-br br0
sudo ovs-vsctl add-port br0 eth1
sudo ovs-vsctl add-port br0 eth2
sudo ovs-vsctl add-port br0 eth0
sudo ovs-vsctl add-port br0 eth4
```

```
sudo ovs-vsctl show
```

Añadimos los flujos correspondientes, que, para este ejemplo, con los correspondientes puertos en el Open vSwitch y direcciones son:

```
sudo ovs-vsctl -- --columns=name,ofport list Interface
```

Flujos relativos a la entrada de tráfico por la interfaz “eth1” (puerto 1):

```
sudo ovs-ofctl add-flow br0 in_port=1,nw_dst=192.168.2.6/24,actions=output:2
sudo ovs-ofctl add-flow br0 in_port=1,dl_type=0x0800,nw_proto=6,actions=output:3
sudo ovs-ofctl add-flow br0 in_port=1,dl_type=0x0800,nw_proto=17,actions=output:4
```

Flujos relativos a la entrada de tráfico por la interfaz “eth2” (puerto 2):

```
sudo ovs-ofctl add-flow br0 in_port=2,nw_dst=192.168.1.5/24,actions=output:1
sudo ovs-ofctl add-flow br0 in_port=2,dl_type=0x0800,nw_proto=6,actions=output:3
sudo ovs-ofctl add-flow br0 in_port=2,dl_type=0x0800,nw_proto=17,actions=output:4
```

Flujos relativos a la entrada de tráfico por la interfaz “eth0” (puerto 3):

```
sudo ovs-ofctl add-flow br0 in_port=3,nw_dst=192.168.1.5/24,actions=output:1
sudo ovs-ofctl add-flow br0 in_port=3,nw_dst=192.168.2.6/24,actions=output:2
```

Flujos relativos a la entrada de tráfico por la interfaz “eth4” (puerto 4):

```
sudo ovs-ofctl add-flow br0 in_port=4,nw_dst=192.168.1.5/24,actions=output:1
sudo ovs-ofctl add-flow br0 in_port=4,nw_dst=192.168.2.6/24,actions=output:2
```

Para hacer persistentes estos cambios:

```
filetool.sh -b
```

Por último, la configuración del nodo “ovs-3”:

Creamos el *bridge* y le añadimos las interfaces correspondientes:

```
sudo ovs-vsctl add-br br0
sudo ovs-vsctl add-port br0 eth5
sudo ovs-vsctl add-port br0 eth4
sudo ovs-vsctl show
```

Añadimos los flujos correspondientes, que, para este ejemplo, con los correspondientes puertos en el Open vSwitch y direcciones son:

```
sudo ovs-vsctl -- --columns=name,ofport list Interface
```

Flujos relativos a la entrada de tráfico por la interfaz “eth5” (puerto 1):

```
sudo ovs-ofctl add-flow br0 in_port=1,actions=output:2
```

Flujos relativos a la entrada de tráfico por la interfaz “eth4” (puerto 2):

```
sudo ovs-ofctl add-flow br0 in_port=2,actions=output:1
```

Para hacer persistentes estos cambios:

```
filetool.sh -b
```

NOTA: En este escenario, también se ha probado y es interesante el siguiente flujo, con el que podemos descartar determinados tráficos.

```
ovs-ofctl add-flow <bridge> <match-field>actions=drop
```

NOTA: Tanto en esta, como en todas las configuraciones, hay que tener presente que son ejemplos. Por tanto, si se quieren replicar hay que tener muy en cuenta los puertos de entrada y salida, además de las direcciones MAC de las máquinas en cada caso.

ANEXO 7 – COMANDOS ADICIONALES

Todos los comandos utilizados para este Trabajo Fin de Grado están detallados en el resto de los Anexos.

Al haber trabajado con equipos con base Linux, se puede encontrar más detalle de los comandos utilizados en sus respectivos manuales. Esta información también puede ser consultada mediante el comando "-h" en el terminal.

Destacamos la página web:
https://docs.pica8.com/display/picos292cg/PicOS+Open+vSwitch+Command+Reference debido a que ha sido consultada en numerosas ocasiones y se considera que posee unas buenas explicaciones y tablas resumen, con ejemplos de todos los comandos. En concreto, resaltamos el apartado de "add-flow" para añadir todo tipo de tráfico: https://docs.pica8.com/pages/viewpage.action?pageId=3083175

NOTA: Una vez introducidos flujos en los *switch,* no resulta trivial consultar el contenido de las tablas donde se almacenan. Para ver estos flujos, hemos decidido que la mejor opción ha sido crear un fichero en blanco (pueba.txt) y posteriormente comparar los comandos introducidos en br0 con nuestro fichero en blanco. De este modo, la respuesta serán todos los comandos de br0, ya que nuestro fichero está vacío. http://manpages.ubuntu.com/manpages/trusty/man8/ovs-ofctl.8.html

```
vi prueba.txt
sudo ovs-ofctl diff-flows br0 ./prueba.txt
```