UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

Tráfico multimedia con Mininet en redes SDN

Samuel Burgos López

Tutor: José Oscar Romero Martínez

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2017-18

Valencia, 4 de julio de 2018

Resumen

El trabajo trata el estudio de la distribución de contenidos multimedia en redes definidas por software (SDN). Estas redes realizan la separación del plano de control del plano de datos, centralizando toda la gestión de una red sobre un software que obtiene el nombre de controlador. Para su estudio se utiliza el emulador Mininet, capaz de definir topologías de redes SDN en un único ordenador, ejecutando un sistema operativo completo en cada dispositivo virtualizado. En las topologías creadas se especifican los parámetros necesarios para proceder al envío de tráfico multimedia entre varios dispositivos, pudiendo analizar y estudiar su comportamiento en tiempo real, realizando pruebas y recogiendo medidas sobre los elementos y enlaces que formen la red, llegando a saturar un enlace para poder ver los efectos que esto conlleva.

Resum

El treball tracta l'estudi de la distribució de continguts multimèdia en xarxes definides per software (SDN). Aquestes xarxes realitzen la separació del plà de control del plà de dades, centralitzant tota la gestió d'una xarxa sobre un programari que obté el nom de controlador. S'utilitzarà l'emulador Mininet, capaç de definir topologies de xarxes SDN en un única computadora, executant un sistema operatiu complet en cada dispositiu virtualitzat. En les topologies creades es especifiquen els paràmetres necessaris per procedir al enviament de trànsit multimèdia entre diversos dispositius, podent analitzar i estudiar el seu comportament en temps real, realitzant proves i recopilant mesures sobre els elements i enllaços que formen la xarxa, arribant a saturar un enllaç per poder comprovar els efectes.

Abstract

The Project is about the study of the distributed multimedia data over Software Defined Networks (SDN). These networks make the distinction between control level and data level, centralizing the management of a network on a software called controller. For the study, it is used the Mininet simulator, which defines network SDN topologies in only one computer, executing a total operative system in each virtualized device. On the created topologies, the parameters necessary to proceed sending multimedia traffic between different devices are specified, being able to analyze and study their behavior in real time, making tests and collecting measurements on the elements and links in the network, saturating a link to see the effects that this entails.

Índice

Capítulo 1. Estado del arte

Internet surge como un proyecto a partir de la necesidad de interconectar equipos. Fue desarrollado por la Agencia de Proyectos de Investigación Avanzados de Defensa (DARPA), lo que dio lugar a la aparición de ARPANET. Esta red se basaba en el uso del Programa de Control de Red (NCP), que consiguió conectar dos nodos entre la Universidad de California y el Instituto de Investigación de Stanford a través del uso del NCP. El rápido desarrollo de este proyecto dio como resultado la aparición del Protocolo de Control de Transmisión (TCP) y del Protocolo de Internet (IP) con los que actualmente funcionan las redes. Gracias a este desarrollo hoy en día podemos disponer de multitud de servicios a través de Internet [1].

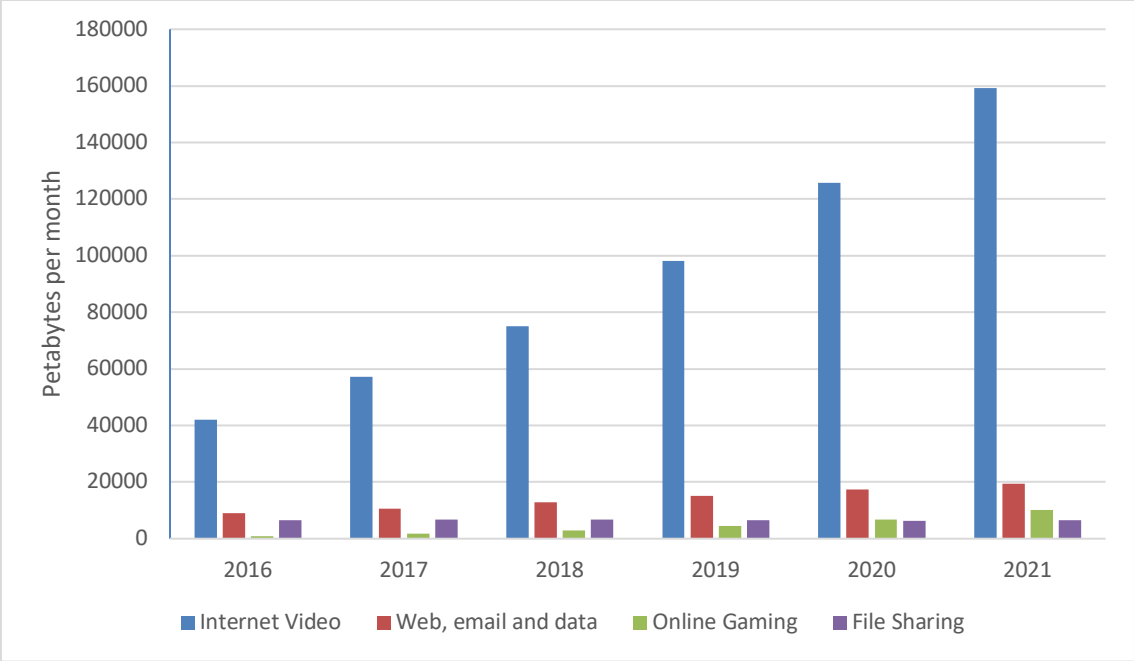
Gracias a las facilidades y las posibilidades que ofrece Internet, se ha producido un gran aumento de dispositivos por usuario que disponen de una conexión a la red. El desarrollo continuo de tecnologías, como la implementación de la Línea de Abonado Digital Asimétrica (ADSL) o Fibra Hasta El Hogar (FTTH), permiten mejorar las características de las redes pudiendo aumentar la velocidad y rendimiento de ellas. Esto conlleva a que cada vez se desarrollen y se oferten más servicios, que anteriormente no se podrían implementar debido a las condiciones que se tenían. La mayoría de servicios tratan de obtener todas las ventajas que ofrecen las redes, exprimiendo estas hasta su límite para ofrecer lo mejor al usuario.

La llegada del concepto de Internet de las Cosas (IoT) ha supuesto una mejora en el día a día de las personas debido a la cantidad de información recogida o a la facilidad de interactuar con cualquier dispositivo que disponga de una conexión a la red. Este aumento de dispositivos conectados, junto con el aumento de dispositivos de uso personal por usuario, incrementan el tráfico que circula por la red, así como la complejidad en la arquitectura de ellas y su gestión para su correcto funcionamiento. Con el gran aumento de tráfico y datos se encuentra el término Big Data, término que se refiere al tratamiento, gestión y procesamiento de cantidades de datos que se van incrementando año a año.

Los servicios en la nube ofrecen múltiples mejoras al usuario, evitando tener que montar una gran red, en la propia red del usuario, y teniendo a su disposición todo lo requerido, como la posibilidad de almacenamiento, bases de datos, procesamiento de datos, etc. Estos nuevos servicios suponen un gran tráfico no solo entre cliente y servidor, sino entre máquina y máquina, ya que no existe un único servidor que almacena toda la información, este tiene que ir obteniendo todos los datos necesarios para dar la respuesta al servicio requerido por el usuario.

Para poder soportar la carga que aportan todos estos servicios, se ha incrementado la velocidad y el tamaño de los servidores. Es posible ver el gran aumento de velocidad en la red que se ha llevado a cabo en un corto periodo de tiempo a través del informe de la CNMC en España, donde la velocidad contratada, en líneas de banda ancha, en el año 2016 contaba con más del 50% de los usuarios por debajo de los 20 Mbps, y en 2017 más del 60% de las líneas supera los 20 Mbps [2].

Analizando el tráfico que circula por la red, el ancho de banda se dispara si se trata de tráfico multimedia. Internet no fue pensada para la distribución de este tipo de tráfico, y junto con el desarrollo de nuevas tecnologías y servicios, aumenta la calidad del contenido multimedia, pasando de contenidos en alta definición (HD), a contenidos en calidades 4K y superiores, provocando que más de la mitad de todo el tráfico que circula por Internet sea tráfico multimedia, en concreto un 71% en 2016, y se prevé que para el año 2021 aumente hasta el 82% [3]. En la Gráfica 1 se puede ver una gráfica con datos y estimaciones realizadas para el tipo de tráfico que se consume en Internet de manera global entre los años 2016 y 2021.



Gráfica 1. Consumo global de tráfico en Internet por segmento

Como se puede apreciar, la cantidad de tráfico de vídeo es muy superior al resto. El ancho de banda utilizado para los servicios que requieran tráfico multimedia es elevado y puede llegar a congestionar la red. El tráfico en la red compite por los recursos disponibles, provocando retardos, *Jitter* y pérdidas. Aumentar el ancho de banda no es siempre una solución, es por ello que se emplea la Calidad de Servicio (QoS), definido por el Sector de Normalización de las Telecomunicaciones, de la Unión Internacional de las Telecomunicaciones (ITU-T), como el efecto colectivo del rendimiento del servicio que determina el grado de satisfacción de un usuario del servicio [4]. Establecer una QoS consiste en cumplir un conjunto de parámetros mínimos, por lo que en caso de no cumplirse estos parámetros se deberá de diferenciar el tráfico y priorizar el establecido.

Centrándonos en el tráfico multimedia, en una red no destinada a un uso exclusivo de tráfico multimedia, este no debe de acaparar todos los recursos disponibles, ya que, en caso de requerir cualquier otro tipo de servicio, como el envío de un correo electrónico, debe de poder realizarse sin problemas. Por ello es importante hablar de la calidad de Experiencia (QoE), que es el grado de aceptación, por parte del usuario final, de una aplicación o servicio [5]. La QoE es subjetiva, e incluye todos los elementos entre ambos extremos, como la red o la infraestructura, pero en cuanto a servicios multimedia se refiere se debe tener muy en cuenta, ya que el usuario tolera cierto rango en los parámetros de QoS por el cual puede visualizar un contenido multimedia. Si sobrepasa ese rango, el contenido puede llegar a ser molesto. Con ello existe una gran diferencia al resto de servicios.

Una solución aportada para evitar la excesiva carga de los servidores, y rebajar la cantidad de tráfico máquina a máquina generado, es la implantación de los sistemas de distribución de contenidos (CDN), en el cual se crean servidores, con el mismo contenido que el servidor centralizado, pero en el extremo de la red.

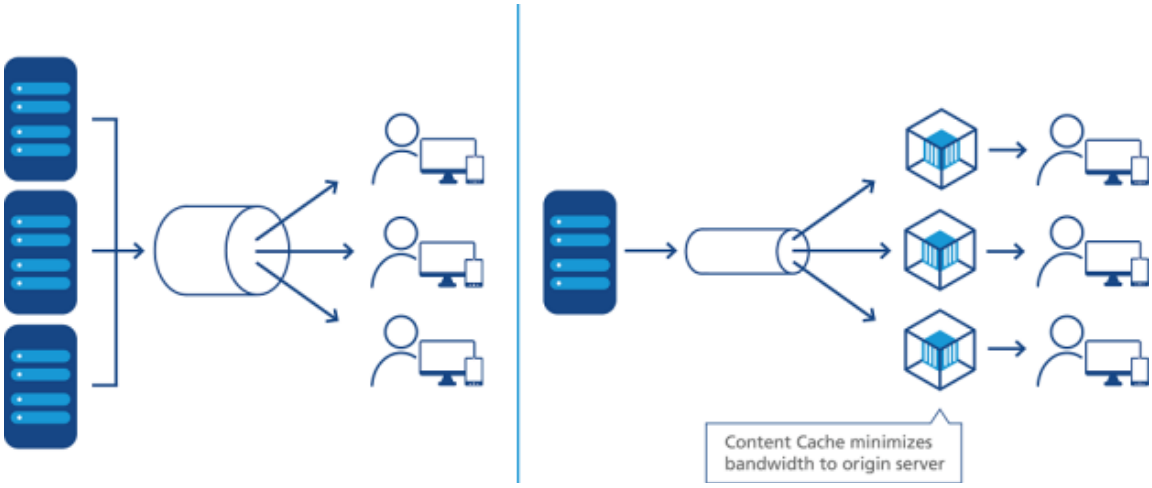


Figura 1. Izquierda: sistema tradicional de distribución de datos. Derecha: sistema de distribución de contenidos (CDN) [6]

Internet es una red abierta, no está gestionada y por lo tanto no existen garantías de QoS. Para poder ofrecer determinados servicios estableciendo una QoS están las arquitecturas de red gestionadas o cerradas. En estas redes se permite el control y la gestión del tráfico para priorizar. También se conocen a los usuarios, a diferencia de Internet en el que el contenido es accesible por cualquier usuario de la red.

Teniendo ya una red gestionada, podremos aplicar una QoS determinada y aumentar el ancho de banda de los enlaces, lo que supone un gasto económico, para evitar problemas de congestión en los enlaces.

Para enviar tráfico de un punto a otro será necesario disponer de enrutadores. Estos tomarán las decisiones de reenvío a través de diferentes protocolos. En la actualidad existen multitud de protocolos de enrutamiento que permiten obtener el mejor camino desde un dispositivo a otro, minimizando tanto el número de saltos como el retardo de propagación. Estos protocolos de enrutamiento dinámicos toman sus decisiones basados en el estado actual de la red, variando las decisiones de enrutamiento en el momento que se produzca un cambio en ella, pero actúan independientemente, complicando la administración y configuración de la red. Es por ello que aplicar una QoS en una red puede llegar a tener una gestión complicada, teniendo en cuenta el crecimiento constante de las redes, lo que supone un gran problema de escalabilidad.

Las redes definidas por software (SDN) surgen para unificar todo el control de la red en un sistema centralizado, facilitando la gestión de la red, pudiendo automatizar y controlar en todo momento los flujos de tráfico, y reduciendo los costes que supondría realizar la misma tarea con un sistema de control distribuido como en la actualidad, proponiendo así una solución para la el mayor aprovechamiento de recursos de red.

1.1 Objetivos

El propósito de este proyecto es estudiar y comprobar el funcionamiento de las redes SDN. Para ello se hará uso del emulador Mininet, creando un entorno de trabajo personalizado en el que se emule una red completa.

El entorno creado será inicialmente de una red sencilla donde se realizarán las pruebas necesarias para comprobar que existe conectividad entre ellas y que se cumplen los parámetros deseados establecidos en la configuración de la red. Una vez creada la red se aumentará la complejidad de esta hasta alcanzar la red deseada.

Para ver el comportamiento de la red SDN, se procederá al envío de flujos de tráfico multimedia entre diferentes dispositivos que formen la red. En ellos se podrán realizar pruebas sobre los distintos parámetros de calidad de servicio, ancho de banda, retardo, *Jitter* y tasa de pérdidas, pudiendo también valorar la calidad de experiencia del tráfico multimedia en tiempo real.

Capítulo 2. Redes definidas por software

2.1 Introducción

Las redes tradicionales no están preparadas para cubrir todos los requisitos de los usuarios y empresas de hoy en día. El control y la gestión de una red puede llegar a ser muy complejo. Una red debe ser capaz de soportar las necesidades que se deban de cubrir, pero debido a los continuos cambios en las necesidades de los usuarios, se debe de estar continuamente supervisando el correcto funcionamiento de la red y adaptándola a las necesidades del momento. Esta tarea puede llegar a ser muy compleja, ya sea por el tamaño de la red o por las necesidades requeridas.

Esta arquitectura de redes proviene de la investigación y el desarrollo de una solución a los problemas vistos anteriormente. Con ello se funda la organización sin ánimo de lucro Open Networking Foundation (ONF), cuyo objetivo es la transformación de la arquitectura de red, promocionando y adoptando la arquitectura SDN, creando estándares para su desarrollo [7].

2.1.1 Definición

Según la definición del Grupo de Trabajo de Ingeniería de Internet (IETF), las redes definidas por software son un conjunto de técnicas usadas para facilitar el diseño, implementación e implantación de servicios de red de una manera determinista, dinámica y escalable [8].

Proporcionan una nueva arquitectura de redes donde el control de la red es completamente programable y está separado del reenvío de paquetes.

2.1.2 Objetivos

El desarrollo de las redes SDN tiene como objetivo proporcionar interfaces abiertas para habilitar el desarrollo de software para controlar un conjunto de recursos de red, pudiendo obtener información sobre el estado y modificar el tráfico en ella.

Con ello se facilita la gestión de la red, centralizando todo el control en un único software, teniendo a disposición del usuario una red altamente escalable y con un control que ofrece una alta granularidad del tráfico que circule a través de ella. Además, supone una mejora de seguridad, gracias a la mejora en el control y la gestión, y la posibilidad de ofrecer, de una manera más rápida, nuevos servicios de red, independientemente del software que posea cada equipo de red, lo que provocará una mejor experiencia para el usuario final.

2.2 Arquitectura

La arquitectura de estas redes define la forma de combinar una red y un sistema informático a través de un software abierto y un hardware que separe el plano de control del plano de datos [9]. Para lograrlo se realiza una separación entre el plano de datos y el plano de control.

El plano de control es el encargado de establecer las rutas de encaminamiento que deben seguir los distintos paquetes de la red. Para ello se basa en información establecida estáticamente o a través del intercambio de información entre los diversos dispositivos de red para calcular las rutas óptimas. En función del intercambio de mensajes se obtiene la arquitectura actual de la red, recalculando rutas, siempre que lo permita el protocolo, en caso de cambios en la red. Estos paquetes intercambiados, son procesados por el dispositivo de red para actualizar la información de la tabla de encaminamiento.

El plano de datos es el encargado de reenviar hacia la interfaz correspondiente cada paquete que entra al dispositivo. Para ello sigue la información aportada por el plano de control, que se encuentra en la tabla de encaminamiento.

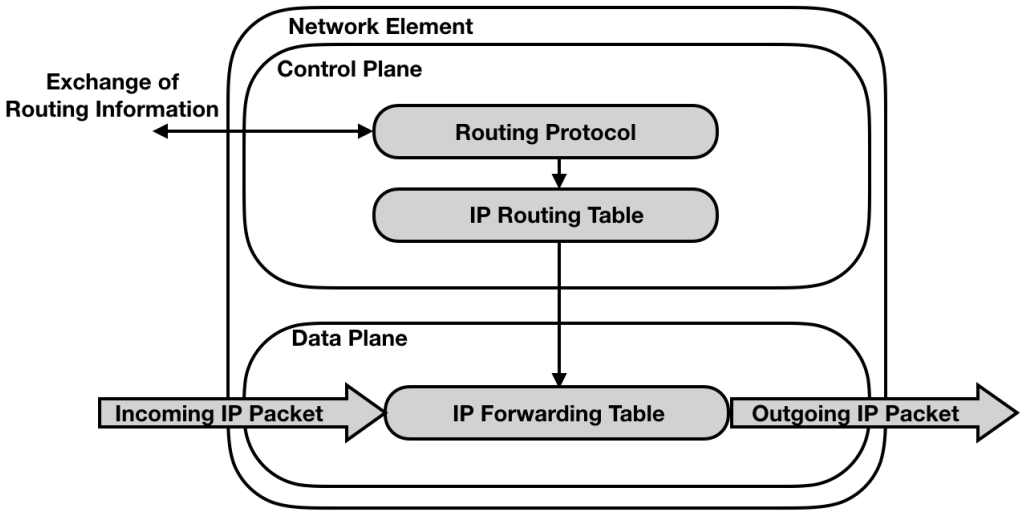


Figura 2. Arquitectura de un dispositivo de red tradicional

En una red tradicional, tanto el plano de datos como el plano de control están en el mismo dispositivo, como se puede apreciar en la Figura 2, pero en una arquitectura de red SDN, el plano de control, pasa a ser externo a los dispositivos de red. Es el elemento llamado controlador, basado en un software, directamente programable. Esto implica que el usuario podrá obtener un control, independientemente del proveedor de los dispositivos de red, desde un único punto, permitiendo simplificar la gestión.

Los dispositivos de red, en redes SDN, simplifican su operación y complejidad, debido a que ya no es necesario procesar ni tener configurados protocolos estándar, si no que se rigen por las órdenes recibidas desde la capa de control.

Uno de los aspectos más importantes es la capacidad de programar las decisiones, no únicamente de forma estática, sino pudiendo aprovechar la inteligencia centralizada del controlador. Con ello se podrá alterar, en tiempo real, el comportamiento de la red, suponiendo una gran mejora en cuanto a tiempo respecto de las redes tradicionales. Gracias a esto es posible la implementación y despliegue de nuevas aplicaciones y servicios en cortos periodos de tiempo, de una forma mucho más sencilla.

Gracias a la mejora de la administración y gestión de la red, es posible aumentar significativamente la protección y seguridad de ella. Al tener un control centralizado de toda la red, se obtiene una visión mucho más clara de qué está pasando, por lo que es posible implementar con más rapidez medidas de seguridad o medidas para optimizar el uso de la red.

Para la comunicación entre las aplicaciones y el controlador, la ONF está investigando interfaces de programación de aplicaciones (APIs) públicas para promover su uso y mejorar la optimización tanto de las aplicaciones como de la red.

Viendo todo el conjunto de una red SDN, su arquitectura se divide en cuatro capas:

- La capa de infraestructura contiene los elementos de red necesarios para el transporte del tráfico de la red. Esta capa actúa como plano de datos tomando sus decisiones según las ordenes recibidas por el controlador. La comunicación entre el controlador y los dispositivos de red se lleva a cabo a través de una interfaz llamada *Data-Controller Plane Interface* (D-CPI).

- En la capa de control encontramos el elemento fundamental de una red SDN, el controlador, que a través de los requisitos de las aplicaciones toma decisiones, previamente programadas, para que la capa de infraestructura pueda satisfacer las necesidades, cumpliendo una serie de políticas instaladas. La comunicación entre la capa de control y la capa de aplicación se lleva a cabo mediante una interfaz llamada *Application-Controller Plane Interface* (A-CPI).

- La capa de aplicación comprende las aplicaciones que interactúen con la red. Estas aplicaciones se comunican con la capa de control mediante la interfaz A-CPI.

- La capa de administración deberá asignar los recursos necesarios y establecer la comunicación entre las distintas capas de aplicación, control y datos, para permitir el funcionamiento de toda la red.

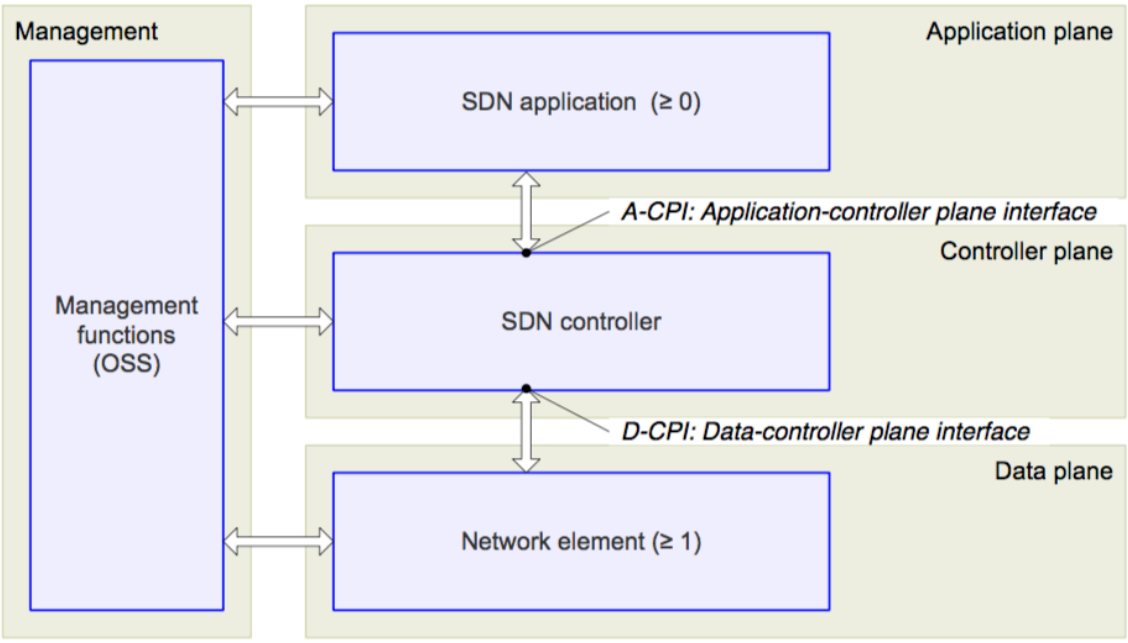


Figura 3. Arquitectura SDN [10].

Una vez vista la arquitectura a nivel alto, se pasa a ver en detalle el contenido de cada capa y los elementos que la forman.

2.2.1 Capa de infraestructura

En esta capa se encuentran todos los dispositivos de red con los recursos necesarios para el reenvío y transporte del tráfico de la red. Estos dispositivos permitirán realizar todas las funciones de red necesarias, ya sea transporte y reenvío de paquetes, como aplicar una QoS, monitorizar el tráfico o realizar acciones de filtrado.

Dentro de cada dispositivo de red encontramos los siguientes componentes:

- Un motor de procesador de tráfico, encargado de realizar el reenvío de los paquetes. Los paquetes deberán pasar a través de este componente, que es el que muestra las decisiones de encaminamiento tomadas.

- Un virtualizador, que abstrae los recursos al controlador.

- Una base de datos de recursos maestros (RDB) en la cual está toda la información de recursos que obtiene el dispositivo de red.

- Un coordinador, que es el encargado de asignar los recursos a los agentes y establecer las políticas especificadas por el usuario desde la capa de administración.

- Uno o más agentes, cuya tarea es aplicar las órdenes recibidas por el controlador SDN. Además, puede compartir recursos o virtualizarlos dependiendo de las necesidades de la red o aplicación, pudiendo aislar los servicios entre clientes diferentes.

En cada dispositivo de red puede haber varios agentes simultáneamente, que son los que establecerán la comunicación con el controlador a través de la D-CPI, también llamadas a menudo Interfaces hacia el sur (SBI). En cuanto al coordinador, únicamente puede existir uno debido a que solo existe una capa de administración. Esto es aplicable tanto en la capa de infraestructura como en la capa de control.

Las funciones de reenvío realizadas en esta capa son tomadas en la capa de control, basadas en función de cómo esté programado el controlador. Es posible asignar que la capa de infraestructura tome decisiones de reenvío autónomas, de esta forma es posible evitar tener que tratar específicamente ciertas funciones, por ejemplo, el tratamiento del Protocolo de Mensaje de Control de Internet (ICMP).

Si en algún momento se recibe un paquete que no esté especificado su tratamiento en la tabla de encaminamiento del dispositivo, este será enviado al controlador para que tome la decisión basada en cómo esté programado.

Para el correcto funcionamiento de la red, estos elementos deberán enviar constantemente información al controlador sobre el estado del elemento o del paquete de tráfico que reciba, ya que con esta información el controlador tomará las decisiones que transmitirá a los elementos de la red. Esta comunicación se lleva a cabo mediante la interfaz D-CPI.

Sobre las tecnologías que se pueden implementar en esta capa, se permite realizar una programación sobre el plano de datos con cualquier tecnología, como Ethernet, IP, OTN, etc.

2.2.2 Capa de control

El elemento principal en la arquitectura de una red SDN es el controlador, este debe estar lógicamente centralizado. Este es una entidad lógica que gestiona el control de toda la red controlando las aplicaciones, a través de una A-CPI, y los dispositivos de red, a través de un protocolo de una D-CPI.

Por lo general, existe un único controlador que ejerce toda la función, aunque podrían existir varios, con procesos idénticos, que realizarían la función de copia de seguridad, en caso de fallo, o la posibilidad de distribuir la carga entre ellos. También se ve la posibilidad de que realizaran procesos distintos, aunque se debe establecer una configuración previa para la colaboración entre los diferentes controladores. Esta configuración debe estar sujeta a una de las siguientes afirmaciones [10]:

- Controlar conjuntos disjuntos de recursos o acciones.

- Establecer una sincronización para que no se pueda dar la situación de emitir órdenes contradictorias o inconsistentes.

Como ejemplo de este caso, encontramos el existe HyperFlow, que se ejecuta entre el plano de control y el plano de infraestructura, el cual ha sido diseñado para tener varios controladores funcionando conjuntamente y evitar problemas de congestión. HyperFlow obtiene las órdenes de cada control de una forma individual, esto permite minimizar el tiempo de respuesta del plano de infraestructura a estas órdenes dadas por los controladores. Además, es sensible a fallos de componentes, ofreciendo soluciones sin que deje de funcionar la red.

Aunque el diseño de un controlador no está especificado, si que hay una serie de componentes mínimos que deben de existir:

- La funcionalidad del plano de datos (DPFC). Este componente posee los recursos necesarios, usados según las órdenes del coordinador o virtualizador. Debe operar, normalmente, como orquestador, ya que el controlador tomará decisiones a varios dispositivos de red.

- Coordinador, que establecerá las configuraciones al resto de componentes a través de las órdenes recibidas desde la capa de administración.

- El virtualizador, proceso que recibe las solicitudes de las aplicaciones, y las valida según las políticas y recursos que han sido asignados a través del controlador, haciéndolas llegar a la interfaz D-CPI y a la DPFC. Es uno de los componentes, junto con la DPFC que interpretan las solicitudes y el intercambio de recursos de los servicios del proveedor.

- Uno o más agentes, que serán los encargados de recibir y enviar las comunicaciones que ocurran entre la capa de aplicación y la capa de control, utilizando la interfaz A-CPI.

2.2.3 Capa de aplicación

Consta de diferentes aplicaciones que mantienen una comunicación con el controlador. Estas aplicaciones especificarán los recursos y el comportamiento requerido por la red. La comunicación se lleva a cabo a través de una A-CPI.

Esta interfaz es la encargada de la comunicación entre la capa de aplicación y la capa de control. Como se ha visto en el punto anterior, se lleva a cabo a través de una A-CPI, también llamadas interfaces hacia el norte (NBI), que especifica la forma de interactuar entre los sistemas. Una de las interfaces más comunes usadas para comunicar esta capa de aplicación con el controlador es el uso de la transferencia de estado representacional (REST), cuya arquitectura se basa en el uso del protocolo de transferencia de hipertexto (HTTP). La ventaja del uso de una interfaz REST, frente a otras opciones, es la escalabilidad y su mantenimiento, ya que hay una separación entre la interfaz de usuario del servidor y los datos almacenados [11].

El uso de una A-CPI también permite la integración con la automatización y la orquestación para que los desarrolladores no deban de preocuparse por el funcionamiento interno de la red, sino únicamente en los requisitos de la aplicación. Actualmente la ONF está investigando el desarrollo de una A-CPI SDN entre la capa de aplicación y la capa de control para intentar encontrar una estandarización [12].

Dentro de cada dispositivo de red encontramos los siguientes componentes:

- La lógica de la aplicación SDN

- Un único coordinador.

- Instancias de las A-CPI necesarias para los agentes que se encuentren en la capa de control.

2.2.4 Capa de administración

Es necesario establecer la configuración del entorno y los parámetros necesarios para la inicialización, establecer la comunicación entre las diferentes capas, estableciendo las credenciales, las políticas a implementar y los protocolos que se van a utilizar.

El controlador puede necesitar información acerca de las adyacencias del plano de datos en sus puertos de red externos. También es posible modificar o establecer los recursos que van a ser utilizados, ya sea por interés del usuario o por necesidad del controlador.

Esta capa tiene más o menos importancia dependiendo del nivel de confianza que exista entre el cliente y el proveedor, donde en algunos casos será mínima la configuración que haya que realizar y en otros será necesario una administración continua para el buen funcionamiento de la red, aunque con el paso del tiempo se irá reduciendo el nivel de configuración necesaria.

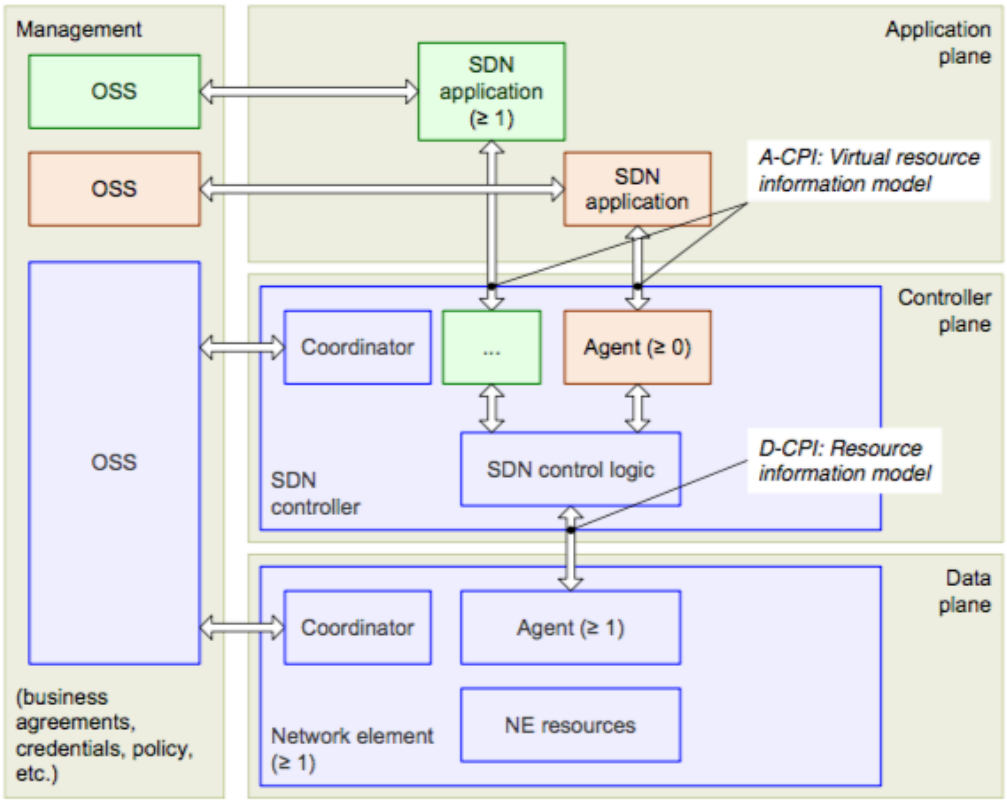


Figura 4. Arquitectura de redes SDN extendida [10]

2.3 OpenFlow

Es la primera interfaz D-CPI estándar definida para la comunicación entre los equipos de red y el controlador. OpenFlow surge a partir de un proyecto en la Universidad de Stanford, en el año 2008. Con la fundación de la ONF, el control de la especificación de OpenFlow pasó a ser supervisado por ella, dando lugar a la versión 1.2, cuyas mejoras más significativas son la posibilidad de permitir la conexión de varios controladores a un único dispositivo de red, soporte para IPv6 [13].

OpenFlow permite tener un acceso directo y manipular los distintos dispositivos de red, ya sean físicos como virtuales basados en hypervisor. Utiliza el concepto de flujos para poder

identificar todo el tráfico que circula por el dispositivo de red, basándose en reglas de coincidencia, que pueden estar programadas estáticamente o dinámicamente, por el controlador. También permite especificar la forma de cómo tratar el tráfico basándose en parámetros como patrones de uso, recursos en la nube y aplicaciones.

Los mensajes que se utilizan en la comunicación entre controlador y conmutador contienen una cabecera están divididos en tres grupos [14]:

- Mensajes simétricos: son enviados tanto por parte del controlador como por parte de los conmutadores. En este grupo se encuentra el mensaje HELLO, usado en la fase de conexión para la negociación de la versión de OpenFlow a usar. El mensaje ERROR, que indica el fallo de una operación que se ha intentado llevar a cabo, como por ejemplo la negociación de la versión. Los mensajes ECHO_REQUEST y ECHO_REPLY son usados para mantener activa la comunicación entre controlador y conmutador, también proporcionan información acerca del ancho de banda y latencia. El mensaje VENDOR, pasado a ser llamado EXPERIMENTER en las versiones de OpenFlow superiores a la 1.0, creado con uso experimental para futuras extensiones.

- Mensajes asíncronos: enviados desde el conmutador al controlador en caso de no encontrar coincidencias de paquete, función que realiza el mensaje PACKET_IN, por expirar alguna entrada en una tabla de flujo, realizado por el mensaje FLOW_REMOVED, o por el cambio del estado en un puerto del conmutador, informado a través del mensaje PORT_STATUS.

- Mensajes Controlador/Conmutador: enviados para obtener, solicitar o establecer configuraciones. En cuanto a configuración nos encontramos con diversos comandos que solicitan y reciben diferente información sobre colas, estadísticas y configuración, como GET_CONFIG_REQUEST/REPLY, FEATURES_REQUEST/REPLY, STATS_REQUEST/REPLY, BARRIER_REQUEST/REPLY. Con el mensaje PACKET_OUT, el controlador es capaz de enviar paquetes al conmutador. Para el tratamiento de las entradas de las tablas de flujo se encuentra el mensaje FLOW_MOD, capaz de realizar dichas acciones. Con el mensaje PORT_MOD se puede modificar el estado de un puerto del conmutador.

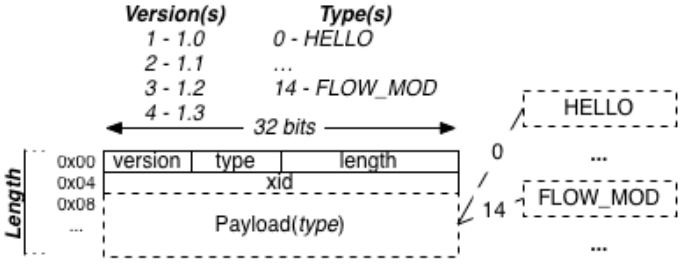


Figura 5. Cabecera común de los mensajes OpenFlow [14]

2.3.1 Conmutador OpenFlow

En caso de no poder basarse en coincidencias ni registros en la tabla, se enviarán los 200 primeros bytes del paquete al controlador para su procesado y la toma de decisión, que será posteriormente transmitida a través de OpenFlow.

Es necesario disponer de un conmutador que pueda trabajar con OpenFlow. Existen tanto conmutadores que únicamente soportan trabajar con redes SDN, como conmutadores con la opción de trabajar tanto con OpenFlow, con otras interfaces o el enrutamiento tradicional.

Un conmutador lógico con OpenFlow contiene tablas de flujo y tablas de grupos. Estas tablas se encargan de realizar búsquedas para encontrar las coincidencias que existan según el tipo de paquete para su reenvío. También contiene uno o más canales OpenFlow, para mantener la comunicación con el controlador, es decir, canales en los cuales se implementará la interfaz D-CPI, en este caso con OpenFlow.

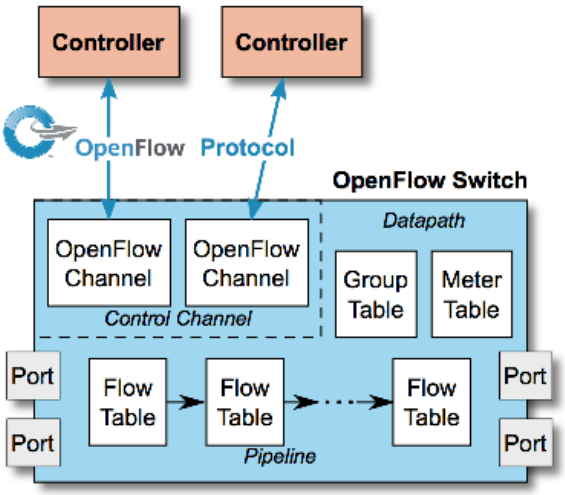


Figura 6. Ejemplo de tabla de instrucciones de OpenFlow [14]

A través del uso de OpenFlow, el conmutador permite modificar, agregar o eliminar las entradas de cada tabla de flujos. Cada tabla contiene los siguientes campos:

- Coincidencia: contiene el puerto de entrada al conmutador y un campo de cabecera. Con ello se buscarán similitudes para tratar cada flujo de paquetes entrantes.
- Prioridad.
- Contador: se actualiza cada vez que se encuentra una coincidencia.
- Instrucciones: indicará el procesamiento del paquete.
- Tiempo de espera: tiempo en el que permanecerá la tabla en el conmutador.
- Cookie: usada por el controlador para el filtrado de entradas. No tiene uso con el procesado de paquetes.
- *Flags*: contiene indicadores que alteran el procesado de paquetes.

Se buscará una coincidencia en la primera tabla de flujo, en caso de no encontrar ninguna coincidencia deberá haber una entrada en la tabla que indique el procesado de los paquetes no coincidentes, pudiendo ser la forma vista anteriormente de reenviar la cabecera del paquete al controlador. Una vez se encuentre coincidencia, en el campo de acciones de la tabla se puede indicar otra tabla en la que se haga otro filtrado más específico, por lo que el paquete irá siendo tratado hasta que no se indique una tabla con la que seguir, en ese momento el paquete se modificará y será reenviado. En este mismo campo de acciones también se incluye el

procesamiento de la tabla de grupos. Esta tabla contiene conjuntos de acciones más complejas, como multirruta, reencaminamiento rápido, agregación de enlaces. Estas tablas contienen los siguientes campos:

- Identificador de grupo: 32 bits sin signo que identifican al grupo de manera única.
- Tipo de grupo: determina la forma de ejecución de los paquetes de acciones
- Contador: actualizado cada vez que los paquetes son procesados por el grupo.
- Paquete de acciones: contiene una lista de las acciones a ejecutar contra los paquetes enviados al grupo.

En la lista de acciones de las tablas de flujo, también es posible especificar una tabla de medición. Estas tablas controlan la tasa del agregado de todas las entradas de flujo a las que está asociada, pudiendo aplicar operaciones simples de QoS, limitando la velocidad, u operaciones más complejas, como DiffServ.

Capítulo 3. Entorno

Lo más importante que debemos tener para poder realizar pruebas en una red SDN, es la capacidad de disponer de la red. Una forma de poder obtener una red SDN para la realización de las pruebas, es de forma virtual, para ello se dispone de varias opciones. Entre los diferentes simuladores y emuladores más usados que permiten el trabajo con redes SDN encontramos el simulador EstiNet [15], el simulador y emulador NS3 [16], y el emulador Mininet. Este último es el elegido para llevar a cabo este trabajo.

A continuación, se va a detallar el software usado para poder construir el entorno y realizar la simulación.

3.1 Sistema operativo

La simulación estará realizada en un sistema operativo Linux con versión 16.04 LTS. En él se dispondrá de los paquetes mostrados a continuación para poder llevar a cabo la simulación completa.

Se utiliza este sistema operativo debido a que el emulador elegido, Mininet, requiere de un *kernel* de Linux, recomendable Ubuntu o Fedora [17], para su funcionamiento. Otra opción sería utilizar una máquina virtual, disponible ya configurada en la propia página del emulador.

3.2 Mininet

Es un emulador de código abierto capaz de crear en una máquina una red virtual realista. El software virtualiza cada dispositivo de la red, controlador, conmutador y enlaces, permitiendo su uso como si se tratara de una máquina. Permite ejecutar el software del sistema donde se ejecute en cada dispositivo emulado [18]. Las redes emuladas ejecutan código sin alterar de los protocolos utilizados en la red, incluyendo el *kernel* de Linux.

Mininet ha sido desarrollado para la investigación de las redes SDN y de OpenFlow. Es de código abierto y posee una alta colaboración para depurarlo e incluir mejoras.

Es posible la creación de diferentes topologías mediante la interfaz de línea de comandos (CLI) integrada en Mininet, además posee una API de Python para experimentación y creación de diferentes redes. Como principales características destaca el rápido prototipado de redes, la creación y testeado de redes complejas sin la necesidad de una red física y la posibilidad de trabajar independientemente en la misma topología de red.

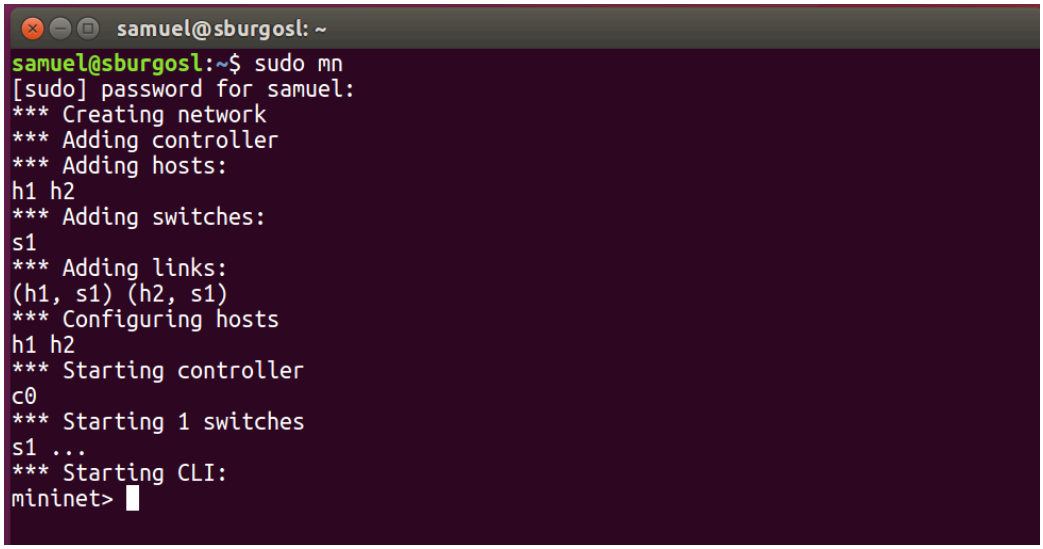


Figura 7. Ejecución Mininet

Para crear una red basta con ejecutar la utilidad `mn` con permisos de administrador. Esto creará una red con un controlador, un conmutador, y dos hosts. La CLI de Mininet provee de una serie de comandos de gran utilidad que nos aportan gran información, como por ejemplo con los comandos, `nodes`, `links`, `dump`, y `net` obtendremos toda la información acerca de la topología y sus interfaces conectadas. Para poder ejecutar comandos dentro de los dispositivos emulados disponemos del comando `xterm`, una vez dentro del dispositivo se podrá realizar todo tipo de acciones soportadas por el sistema operativo, aparte de comandos específicos de conmutación de paquetes, dependiendo del tipo de dispositivo.

Mininet por defecto utiliza OpenFlow para la comunicación entre el controlador y los dispositivos, y emulará conmutadores OpenvSwitch (OVS). Esto es posible modificarlo a través de las opciones a la hora de iniciar la red, con `--switch` y `--controller`.

Para la creación de topologías sencillas basta con añadir la opción `--topo` a la hora de iniciar la red, indicando el tipo de topología, de las que hay predefinidas, y especificando el número de hosts.

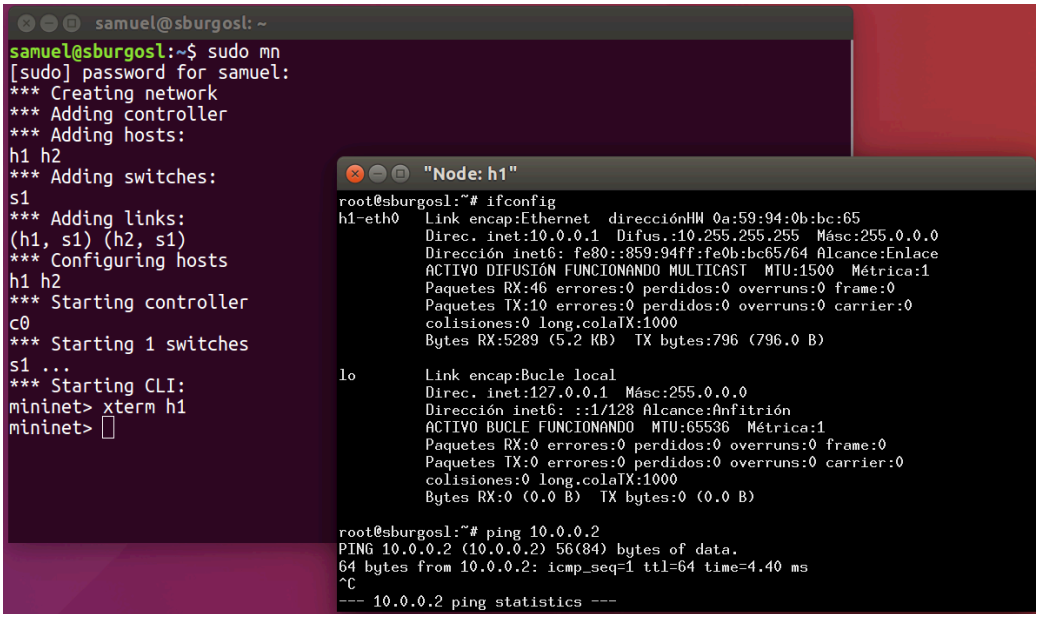


Figura 8. Configuración de interfaces red en un Host de Mininet y ejecución de la utilidad ping ,a través de comandos de Linux.

3.2.1 Miniedit

Es un *script* en Python incluido en Mininet, que al ejecutarse permite, aportando una interfaz gráfica sencilla, añadir los diferentes dispositivos y enlaces para poder generar cualquier tipo de red. Permite personalizar los diferentes dispositivos, configurando IPs, controlador, protocolos, etc. Una vez creada es posible ejecutar dicha simulación.

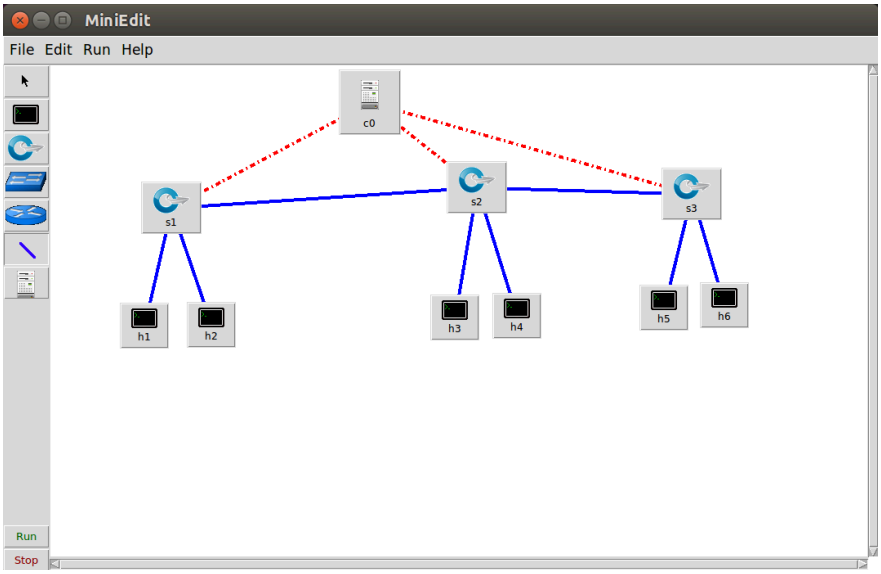


Figura 9. Miniedit

Desde la creación de una red en Miniedit es posible la exportación de dicha red en un *script* de Python. Este *script* es de gran ayuda a la hora de realizar uno propio sin la ayuda de esta interfaz, debido a que incluye varios comentarios y una estructura sencilla para la edición y modificación de la red.

```python
#!/usr/bin/python

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

def myNetwork():

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

    info( '*** Adding controller\n' )
    c0=net.addController(name='c0',
                         controller=Controller,
                         protocol='tcp',
                         port=6633)

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

    info( '*** Add hosts\n')
    h5 = net.addHost('h5', cls=Host, ip='10.0.0.5', defaultRoute=None)
    h2 = net.addHost('h2', cls=Host, ip='10.0.0.2', defaultRoute=None)
    h3 = net.addHost('h3', cls=Host, ip='10.0.0.3', defaultRoute=None)
    h6 = net.addHost('h6', cls=Host, ip='10.0.0.6', defaultRoute=None)
    h1 = net.addHost('h1', cls=Host, ip='10.0.0.1', defaultRoute=None)
    h4 = net.addHost('h4', cls=Host, ip='10.0.0.4', defaultRoute=None)

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

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

    info( '*** Starting switches\n')
    net.get('s3').start([c0])
    net.get('s1').start([c0])
    net.get('s2').start([c0])

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

    CLI(net)
    net.stop()

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

Figura 10. Script en Python generado por Miniedit

3.3 VLC

Se trata de un reproductor multimedia y un *framework* capaz de reproducir la mayoría de archivos y flujos multimedia. Es un software libre, multiplataforma y de código abierto [19]. Posee la capacidad de realizar distribución de contenido a través de la red, *streaming*. Además

En este trabajo será necesario para poder enviar y reproducir el contenido multimedia a través de la red SDN. Con la visualización podremos obtener la variación en tiempo real de la QoE dependiendo de la saturación que exista en el enlace.

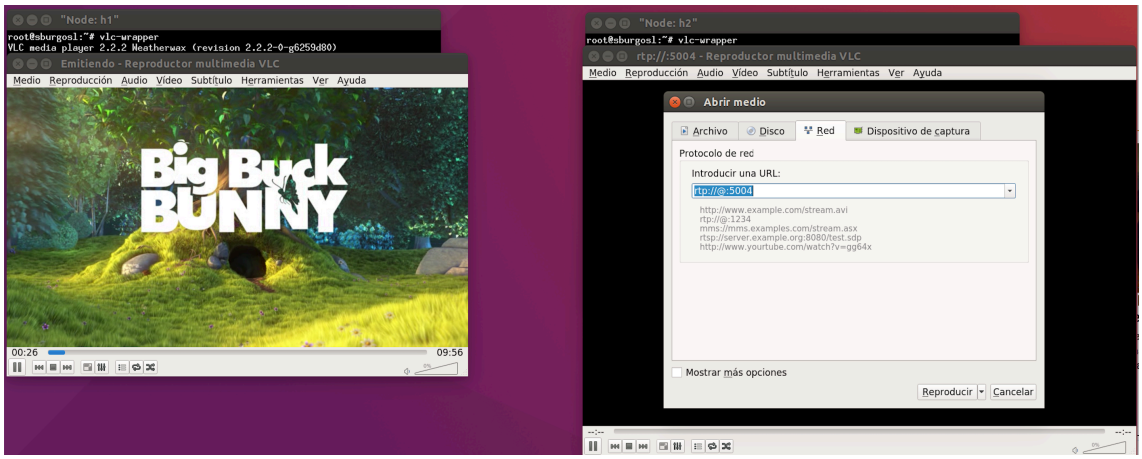


Figura 11. Emisión y recepción de contenido multimedia entre Host en una red SDN

El contenido multimedia será enviado mediante la creación de contenedores MPEG2-TS y su encapsulación sobre el protocolo de transporte en tiempo real (RTP), que a su vez irá sobre el Protocolo de Datagramas de Usuario (UDP) e IP, formando la trama mostrada en la Figura 12.

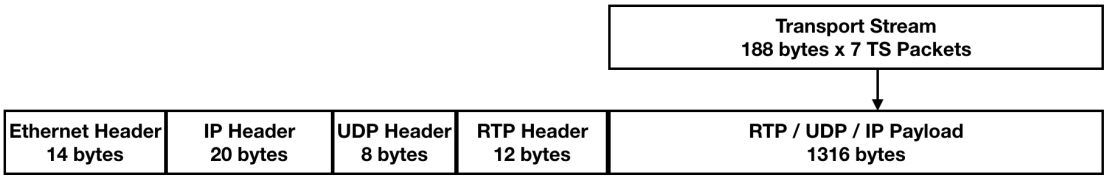


Figura 12. Trama MPEG2-TS encapsulada sobre RTP/UDP/IP

Un ejemplo de un paquete MPEG2-TS recibido, capturado con Wireshark, se puede apreciar en la Figura 13. El tamaño de cada paquete mostrado, 1340 bytes, se corresponde con lo visto anteriormente, al igual que el contenido que transporta RTP, MPEG2-TS, es indicado en la propia cabecera RTP junto con un número de secuencia, que permitirán calcular la tasa de pérdidas.

3.4 Wireshark

Es un analizador de protocolos de red [20]. Permite capturar y analizar de forma interactiva el tráfico que circula sobre una interfaz, mostrando la información de cada campo, y obteniendo los flujos de conexiones completas a través de una interfaz gráfica. Es un programa de código abierto y que, entre otras, permite ejecutarse en sistemas operativos Linux.

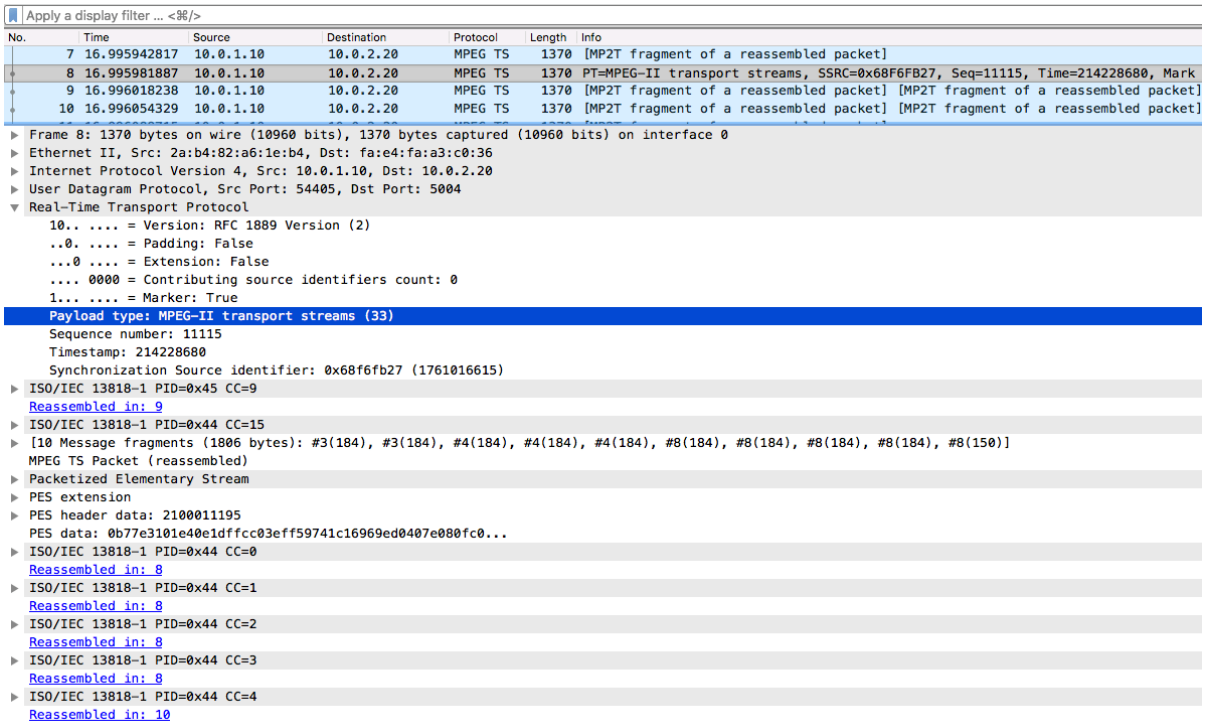


Figura 13. Captura de Wireshark de un paquete MPEG2-TS enviado por VLC en una red SDN

Con el uso de Wireshark podremos capturar los paquetes enviados entre los conmutadores y el controlador y los flujos del protocolo de transporte en tiempo real (RTP) enviados y recibidos por cada interfaz, obteniendo información y estadísticas de los mismos.

El calculo de los paquetes perdidos se realiza gracias al número de secuencia incluido en los paquetes RTP. Este número de secuencia es aleatorio en el primer paquete, incrementándose por unidades en los paquetes sucesivos. El receptor, a través del primer y último número de secuencia calculará los paquetes perdidos en función de los números de secuencia que falten.

3.5 Python

Es un lenguaje de programación interpretado, orientado a objetos e interactivo. Es independiente de la plataforma en la que se ejecute, por lo que únicamente es necesario obtener la versión deseada en el sistema operativo desde el que se vaya a ejecutar. Destaca la clara sintaxis que tiene respecto a otros lenguajes de programación similares, además de disponer de multitud de funciones y librerías incorporadas en el lenguaje, o disponibles para su importación [21].

Toda la red SDN en Mininet estará creada a partir de un código Python. En él estableceremos los dispositivos de red, y los enlaces entre cada uno de ellos.

Un ejemplo de este uso se puede ver en la Figura 10.

3.6 Bash (Bourne-again Shell)

Es un intérprete de comandos para un sistema operativo tipo Unix de software libre GNU. Actúa a su vez tanto de intérprete de comandos como de lenguaje de programación, proporcionando una interfaz y un lenguaje para combinar las dos utilidades, por lo que permite la ejecución de comandos GNU tanto de manera síncrona como asíncrona. Incorporan a su vez un conjunto de comandos para implementar funcionalidades o permitir acceder a algunas de las ya implementadas de una manera más cómoda [22].

Ya que cada dispositivo de red creado en Mininet virtualiza un sistema operativo Linux, será necesario el uso de comandos sobre Bash, ya que es el intérprete de comandos por defecto, para poder ejecutar los diversos paquetes o realizar configuraciones en el dispositivo.

3.7 Iperf

Es una herramienta para realizar mediciones de ancho de banda máximo en redes IP. Es multiplataforma y de código libre, desarrollado por Esnet / Lawrence Berkeley National Laboratory [23]. Funciona tanto con IPv4 como con IPv6, sobre los protocolos de transporte TCP, UDP y el Protocolo de Transmisión de Control de Flujo (SCTP). Con la creación de flujos, entre un Host y un Servidor, obtiene distinta información, dependiendo del protocolo, respecto al ancho de banda. Permite una alta configuración en cuanto a la realización de las mediciones, pudiendo especificar temporizadores, número de intentos, puertos y tamaño de los datagramas, entre otros.

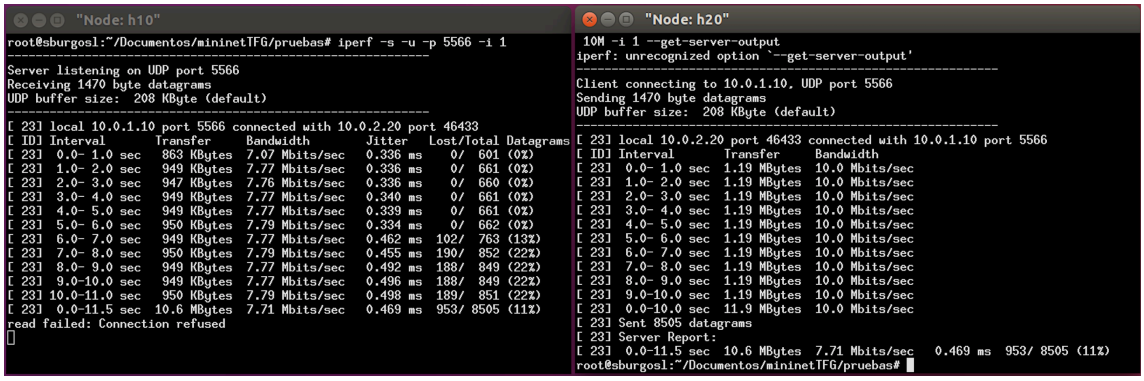


Figura 14. Uso de la herramienta Iperf dentro del Bash de Hosts en una red SDN

3.8 D-ITG

Se trata de una plataforma capaz de generar tráfico IP. Está creado para replicar con precisión la carga de trabajo que realizan las aplicaciones actuales sobre las redes. [24]

Permite generar tráfico siguiendo modelos estocásticos, imitando comportamientos de protocolos a nivel de aplicación en cuanto al tamaño de los paquetes y los tiempos entre salidas.

A parte de generar tráfico, facilita una serie de medidas para evitar tener que depender de otros programas. La información facilitada muestra información general de la prueba, como duración, número de paquetes enviados, velocidad de bits media, pérdidas, paquetes duplicados entre otros, e información respecto a la QoS, como velocidad de bits, *Jitter*, pérdidas de paquetes por segundo y tiempo de ida y vuelta (rtt).

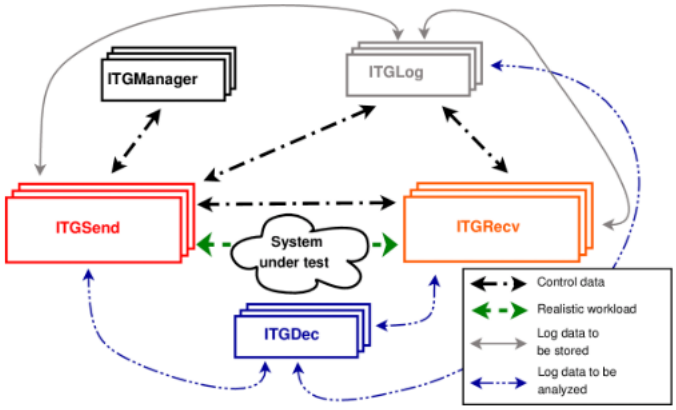


Figura 15. Arquitectura D-ITG [24]

La arquitectura de esta plataforma está dividida en una serie de componentes:

- ITGRecv: es el componente encargado de recibir el tráfico. Se puede establecer la IP desde donde se recibirá el flujo por si a un mismo Host receptor existen varios flujos.
- ITGSend: es el componente encargado de enviar el tráfico a ITGRecv. Mendiante este componente es posible editar puertos, especificar IP del componente ITGRecv, elegir el protocolo de transporte, editar el número de paquetes enviados, ya sea absoluto o por segundo, y el tamaño de los paquetes. Contiene numerosas otras opciones configurables, pero las anteriormente nombradas son las más importantes. También es posible establecer rangos, por ejemplo, del tamaño de paquete, de forma que aleatoriamente varíe en función de ese rango. También permite enviar hacia múltiples receptores flujos de forma paralela.
- ITGLog: es el encargado de almacenar la información recibida.
- ITGManager: es un ejemplo del uso que tendría la API D-ITG para el control del componente ITGSend remotamente.
- ITGDec: este componente decodifica y analiza los archivos de registro almacenados. Permite mostrar la información que será visible al usuario, mencionada anteriormente.

Es importante saber qué información se proporciona y como se obtiene. Cada paquete enviado contiene dos marcas de tiempo, txTime se corresponde del tiempo de transmisión y rxTime se corresponde a el tiempo de recepción. Estas marcas de tiempo pueden variar dependiendo del modo de medida elegida en las opciones del componente ITGSend.

- One-way delay meter (`-m owdm`): es la medida utilizada por defecto. Ambas marcas de tiempo generadas por el componente ITGSend se corresponden al tiempo de transmisión, por lo que el retardo y *jitter* será nulo.
- Round-trip time meter (`-m rttm`): En este caso, para el componente ITGSend, la marca de tiempo de transmisión se corresponde al tiempo de transmisión del paquete y la marca de recepción se corresponde con el instante de tiempo que se recibe la confirmación desde el componente ITGRecv, por lo que el retardo se corresponde al tiempo de propagación de ambos sentidos, y el *jitter* a la variación de tiempos de propagación de ambos sentidos.

El componente ITGRecv mantiene el tiempo de transmisión, txTime, que recibe desde ITGSend, y pone el tiempo de recepción en la marca rxTime. De este modo el retardo se corresponde al tiempo de propagación y el *jitter* a la variación de tiempos de propagación, que

es el dato que se desea. Por ello obtendremos los datos desde el *log* generado por el componente ITGRecv.

El tiempo total mostrado es la diferencia entre el tiempo de recepción del primer y último paquete enviados. El retardo es la diferencia entre el tiempo de transmisión y recepción de cada paquete y la desviación (σ) estándar de retardo viene dada por la ecuación (3.1), donde N es el número de paquetes, d_i es el retardo del paquete i y $\hat{d}$ es la media de retardo los paquetes.

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(d_i - \hat{d}\right)^2} \tag{3.1}$$

El *Jitter* es calculado a partir de una función estimadora de acuerdo a las ecuaciones (3.2), en las cuales S_i es la marca de tiempo de transmisión de cada paquete i, R_i es la marca de tiempo de recepción de cada paquete i y n es el número de D_i calculados.

$$D_i = (R_i - S_i) - (R_{i-1} - S_{i-1}) = (R_i - R_{i-1}) - (S_1 - S_{i-1})$$
$$AvgJitter = \frac{\sum_{i=1}^{n}|D_i|}{n} \tag{3.2}$$

En la siguiente figura se muestra un ejemplo de los momentos en los que se establecen las marcas de tiempo de transmisión y recepción,

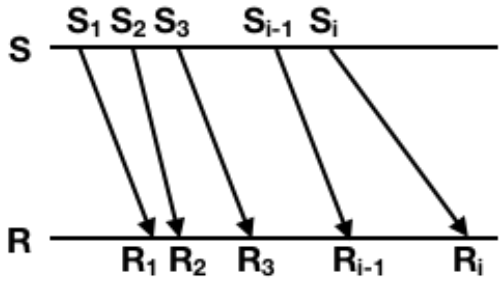


Figura 16. Marcado de tiempos txTime y rxTime de paquetes con D-ITG

Capítulo 4. Pruebas

4.1 Creación del entorno

Para comenzar, lo primero de todo será crear la red por la cual vamos a proceder al envío de tráfico. Con la ejecución de Mininet es posible, a través de añadir unos parámetros, el crear una topología de red SDN. Simplemente con ejecutar Mininet se iniciará una red con una topología de dos hosts, un conmutador y un controlador.

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

Figura 17. Ejecución de Mininet

Para obtener la topología de la red y las interfaces conectadas utilizaremos el comando `net`, que nos proporciona la información mostrada en la siguiente Figura.

```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0
c0
```

Figura 18. Topología por defecto de una red SDN en Mininet

Al iniciar la red, sin personalización, se asignan por defecto unas direcciones IP y Mac. Las direcciones IP coincidirán con el número de host, pero las direcciones Mac se asignan aleatoriamente. Para obtener una facilidad a la hora de realizar configuraciones se incluirá a la hora de ejecutar Mininet la opción `--mac`, de esta forma las Mac asignadas serán más cómodas para trabajar con ellas.

Host	IP	Mac
h1	10.0.0.1	2A:AF:34:91:6A:58
h2	10.0.0.2	46:DF:17:58:4C:31

Tabla 1. Ejemplo direcciones IP y Mac asignadas en Mininet

Host	IP	Mac
h1	10.0.0.1	00:00:00:00:00:01
h2	10.0.0.2	00:00:00:00:00:02

Tabla 2. Ejemplo direcciones IP y Mac asignadas en Mininet con la opción `--mac`

Para poder comprobar la conectividad de toda la red Mininet dispone del comando `pingall` que evita tener que realizar un ping desde cada dispositivo al resto. La salida del comando

mostrará los dispositivos accesibles desde cada host o router, junto con un porcentaje de paquetes ICMP perdidos. En caso de no realizar ninguna modificación a la topología por defecto se obtiene conectividad entre los hosts h1 y h2 de manera bidireccional.

Es importante saber que por defecto Mininet utilizará conmutadores Open vSwitch (OVS), en modo OpenFlow y un controlador que usará OpenFlow para poder administrar el control de la red. Con la misma red por defecto iniciada en Mininet es posible mirar las tablas que tiene el conmutador. Nada más iniciar la red es posible ver las tablas de flujo del conmutador s1, que estarán sin ninguna entrada. Tras la ejecución del comando `pingall` el conmutador OVS se comunicará con el controlador, que con su respuesta insertará las entradas necesarias en la tabla.

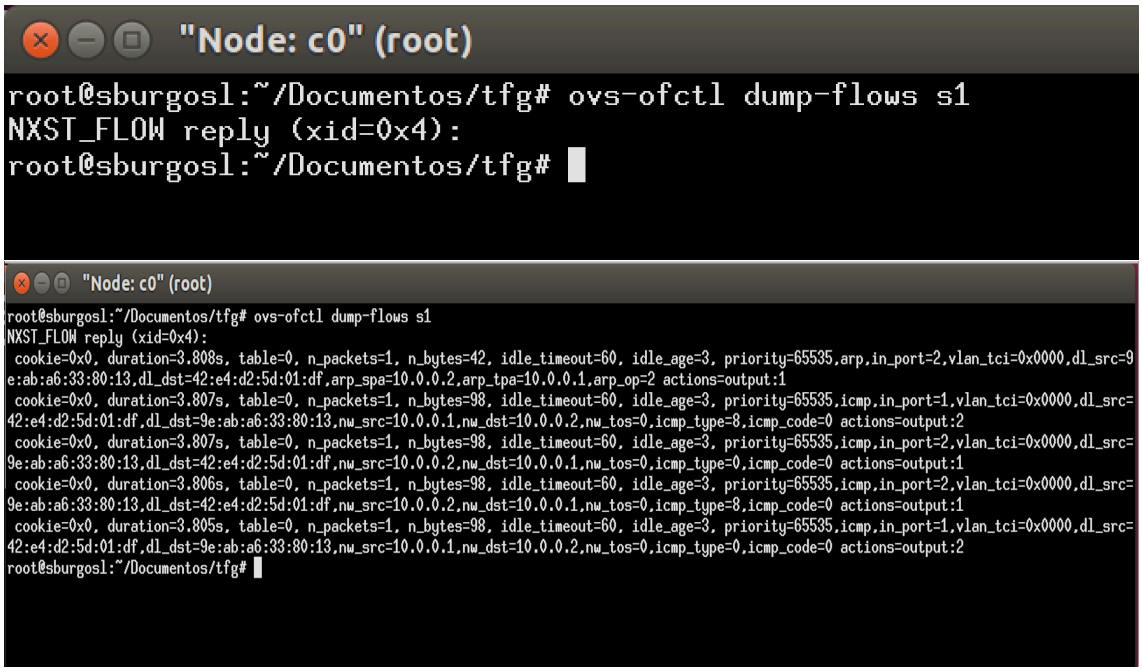


Figura 19. Tablas de flujo en s1. Arriba: Al iniciar la red. Abajo: Una vez realizado `pingall`

Como se puede ver, el controlador ha recibido los mensajes enviados por los conmutadores y les ha mandado las órdenes necesarias para establecer las rutas. Estos mensajes, vistos con más detalle en 2.3,se pueden capturar con Wirehsark.

No.	Time	Source	Destination	Protocol	Length	Info
2	0.140832688	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
3	0.140873137	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
4	0.141738331	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REPLY
6	0.141823291	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REPLY
9	2.140873097	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
10	2.141119174	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REPLY
15	4.339086051	00:00:00_00:00:01	Broadcast	OpenFlow	128	Type: OFPT_PACKET_IN
16	4.339505735	127.0.0.1	127.0.0.1	OpenFlow	92	Type: OFPT_PACKET_OUT
20	4.340130054	00:00:00_00:00:02	00:00:00_00:00:01	OpenFlow	128	Type: OFPT_PACKET_IN
21	4.340441302	127.0.0.1	127.0.0.1	OpenFlow	148	Type: OFPT_FLOW_MOD
24	4.341228772	10.0.0.1	10.0.0.2	OpenFlow	184	Type: OFPT_PACKET_IN
25	4.341542094	127.0.0.1	127.0.0.1	OpenFlow	148	Type: OFPT_FLOW_MOD
28	4.342382927	10.0.0.2	10.0.0.1	OpenFlow	184	Type: OFPT_PACKET_IN
29	4.342618466	127.0.0.1	127.0.0.1	OpenFlow	148	Type: OFPT_FLOW_MOD
32	4.345011311	10.0.0.2	10.0.0.1	OpenFlow	184	Type: OFPT_PACKET_IN
33	4.345133367	127.0.0.1	127.0.0.1	OpenFlow	148	Type: OFPT_FLOW_MOD
36	4.345361188	10.0.0.1	10.0.0.2	OpenFlow	184	Type: OFPT_PACKET_IN
37	4.345479373	127.0.0.1	127.0.0.1	OpenFlow	148	Type: OFPT_FLOW_MOD
41	5.140669654	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
42	5.140704514	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
43	5.140844133	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REPLY
45	5.140898616	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REPLY
52	9.140561291	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
53	9.140782725	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REPLY
56	9.536056430	00:00:00_00:00:02	00:00:00_00:00:01	OpenFlow	128	Type: OFPT_PACKET_IN
57	9.536338251	127.0.0.1	127.0.0.1	OpenFlow	148	Type: OFPT_FLOW_MOD
61	9.536840610	00:00:00_00:00:01	00:00:00_00:00:02	OpenFlow	128	Type: OFPT_PACKET_IN
62	9.537088441	127.0.0.1	127.0.0.1	OpenFlow	148	Type: OFPT_FLOW_MOD
65	10.139908287	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
66	10.139939175	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
67	10.140711418	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REPLY
69	10.140774129	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REPLY

Figura 20. Mensajes OpenFlow generados para permitir la conectividad en la red por defecto

Este controlador por defecto ya está programado. Para mantener un control más exhaustivo de la red se eliminará el controlador por defecto, sustituyéndolo por un controlador remoto sin programar. De esta manera se meterán manualmente, desde el controlador, los comandos OVS, como si se tratase de decisiones tomadas por él, para establecer las tablas de los conmutadores. Para ello se debe añadir como opción en la ejecución de Mininet `--controler remote`.

Para crear una topología más compleja se puede añadir al comando de ejecución de Mininet la opción `--topo=tree,3,2` siendo el 3 el número de capas de conmutadores, y 2 el número de hosts de la capa más baja de conmutadores. Este comando nos creará una red con la topología de la Figura 21.

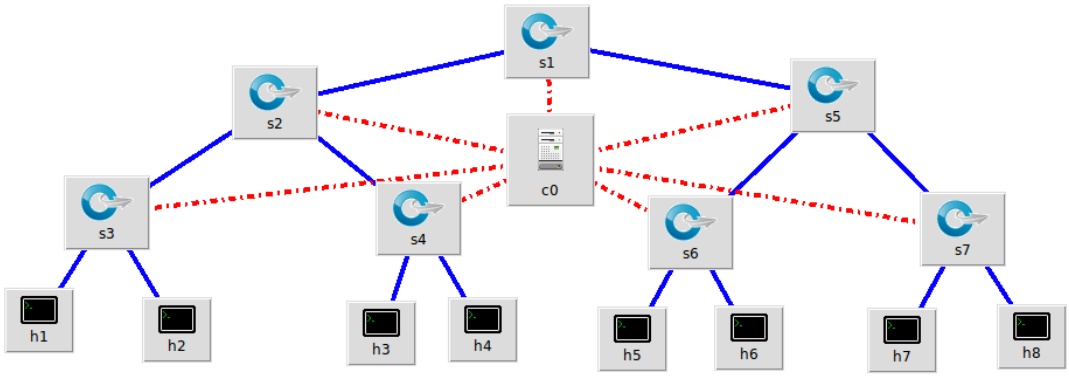


Figura 21. Topología de red en árbol, esquema de Miniedit

En las topologías predefinidas es posible verificar la conectividad, tanto con el controlador por defecto como introduciendo manualmente las órdenes.

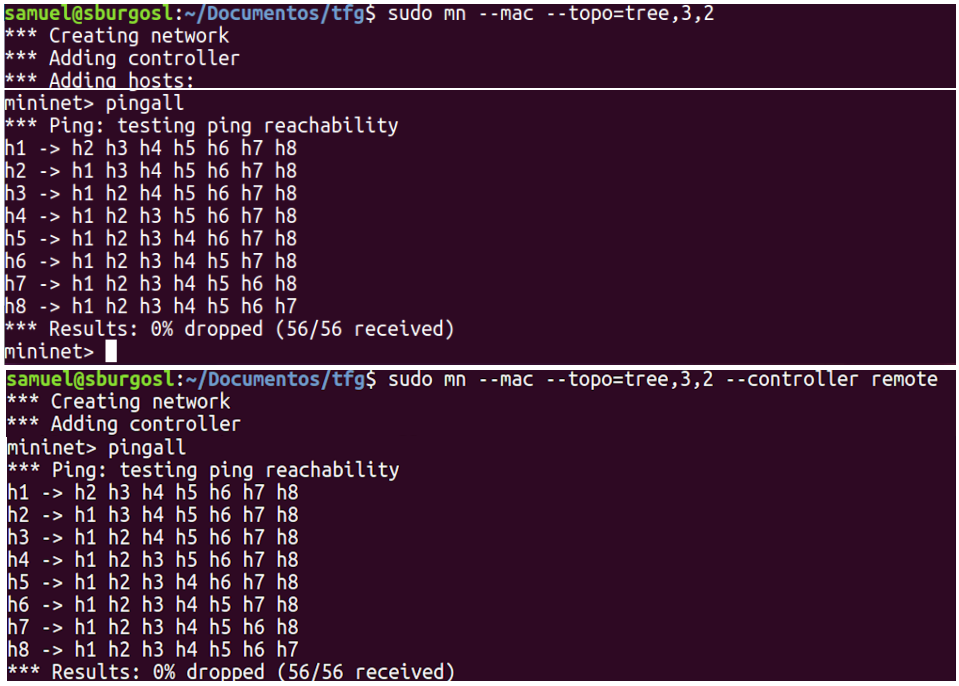


Figura 22. Prueba de conectividad entre todos los hosts de una red con topología en árbol. Arriba: Controlador por defecto. Abajo: Controlador sin programación previa.

Para la creación de una topología más personalizada se utilizará un script de Python. En el script añadiremos los siguientes dispositivos con la configuración que se muestra en la Tabla 3.

Tipo de dispositivo	Nombre	IP	Mac	Interfaces
Host	h21	10.0.2.21/16	00:00:00:00:00:21	h21-eth0
Host	h22	10.0.2.22/16	00:00:00:00:00:22	h22-eth0
Host	h31	10.0.3.31/16	00:00:00:00:00:31	h31-eth0
Host	h32	10.0.3.32/16	00:00:00:00:00:32	h32-eth0
Switch OVS	s1			s1-eth1; s1-eth2
Switch OVS	s2			s2-eth1; s2-eth2; s2-eth3
Switch OVS	s3			s3-eth1; s3-eth2; s3-eth3

Tabla 3. Configuración dispositivos de la red SDN

Los enlaces estarán conectados según a la información mostrada en la Tabla 4.

Velocidad de enlace	Dispositivo A	Interfaz de dispositivo A	Dispositivo B	Interfaz de dispositivo B
Ilimitada[1]	h21	h21-eth0	s2	s2-eth1
Ilimitada	h22	h22-eth0	s2	s2-eth2
Ilimitada	s2	s2-eth3	s1	s1-eth1
Ilimitada	s1	s1-eth2	s3	s3-eth3
Ilimitada	s3	s3-eth1	h31	h31-eth0
Ilimitada	s3	s3-eth2	h32	h32-eth0

Tabla 4. Conectividad enlaces de la red SDN

Con toda la configuración mostrada anteriormente obtendremos el siguiente esquema de red.

[1] Si no se especifica una velocidad, esta será limitada por la potencia del sistema.

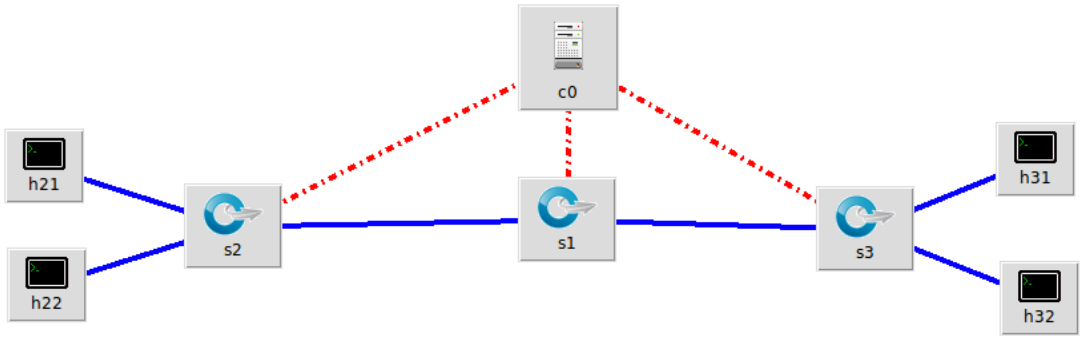


Figura 23. Esquema de la topología creada

Para poder iniciar la red que hemos creado deberemos de utilizar el comando `sudo mn` pasándole los siguientes parámetros:

- `--custom <script_path>`
  Permite leer clases personalizadas desde el fichero que se especifique en la opción.
- `--topo`
  Permite seleccionar, dentro del fichero, la topología deseada para su emulación.
- `--link=tc`
  Permite especificar limites de ancho de banda, pérdidas, retardo de propagación, entre otros parámetros.
- `--controller remote`
  Eliminamos un controlador previamente programado asumiendo manualmente el control de la red.

Finalmente, para comprobar el buen funcionamiento de la red se realiza una prueba de *ping* obteniendo una conectividad completa entre todos los *hosts* como se muestra en la siguiente figura.

```
samuel@sburgosl:~/Documentos/tfg$ sudo mn --custom one_path_TFG.py --topo onepath --link=tc --controller remote
[sudo] password for samuel:
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Unable to contact the remote controller at 127.0.0.1:6633
Setting remote controller to 127.0.0.1:6653
*** Adding hosts:
h21 h22 h31 h32
*** Adding switches:
s1 s2 s3
*** Adding links:
(h21, s2) (h22, s2) (h31, s3) (h32, s3) (s1, s2) (s1, s3)
*** Configuring hosts
h21 h22 h31 h32
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet> xterm c0
mininet> pingall
*** Ping: testing ping reachability
h21 -> h22 h31 h32
h22 -> h21 h31 h32
h31 -> h21 h22 h32
h32 -> h21 h22 h31
*** Results: 0% dropped (12/12 received)
```

Figura 24. Test de conectividad en la topología creada

4.2 Envío de tráfico

4.2.1 Envío de tráfico de relleno en una red sin limitación de ancho de banda

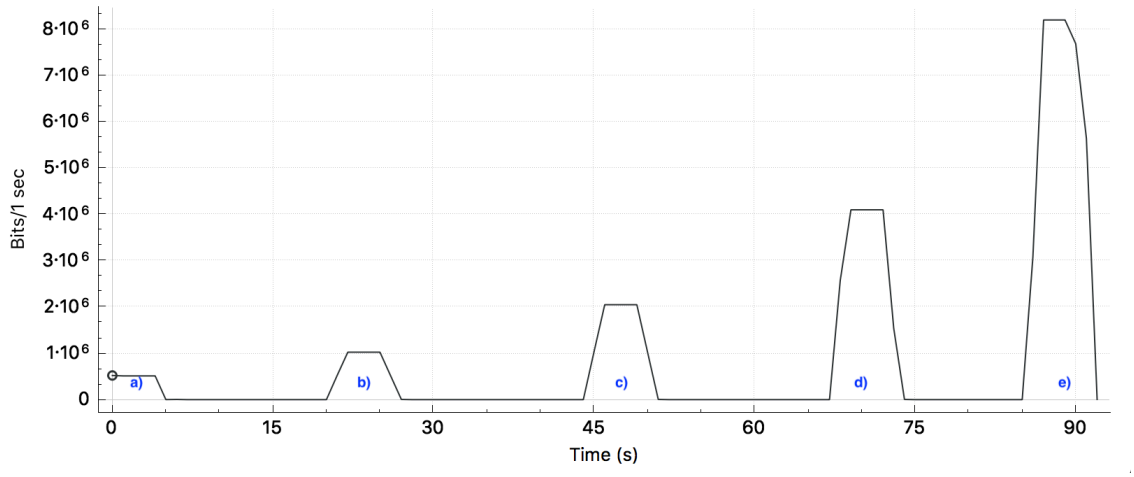
Una vez creada la topología de red deseada procederemos al envío de tráfico. Para ello usaremos la plataforma D-ITG.

Desde el *host* h21 emitiremos tráfico, utilizando el protocolo UDP, usando el componente ITGSend hacia el *host* h31, donde tendremos activo el componente ITGRecv, con los siguientes parámetros:

Prueba	IP Origen	IP Destino	Tamaño paquete (bytes)	Paquetes / Segundo	Duración (ms)	Ancho de banda / Segundo (Mb)
a)	10.0.2.21	10.0.3.31	64000	1	5000	0,5
b)	10.0.2.21	10.0.3.31	64000	2	5000	1
c)	10.0.2.21	10.0.3.31	64000	4	5000	2
d)	10.0.2.21	10.0.3.31	64000	8	5000	4
e)	10.0.2.21	10.0.3.31	64000	16	5000	8

Tabla 5. Datos de configuración para envío de tráfico con D-ITG

Tras realizar las diferentes pruebas de manera continuada podemos obtener, a través de Wireshark, una gráfica del ancho de banda usado en bits/s, en la que también se incluye la prueba a la que corresponde.



Gráfica 2. Ancho de banda medido con las pruebas de envío de tráfico.

Al no haber limitación de ancho de banda la gráfica es igual tanto para la interfaz de h21 como en la interfaz de h31. En la Tabla 6 se pueden ver los datos obtenidos sobre la realización de estas pruebas.

	Prueba a)	Prueba b)	Prueba c)	Prueba d)	Prueba e)
Paquetes enviados	5	10	20	40	80
Retardo mínimo	99 µs	80 µs	68 µs	58 µs	73 µs
Retardo máximo	1996 µs	1809 µs	3610 µs	3151 µs	2720 µs
Media del retardo	506 µs	272 µs	292 µs	249 µs	196 µs
σ del retardo	745 µs	512 µs	762 µs	582 µs	333 µs
Media del Jitter	496 µs	196 µs	210 µs	240 µs	116 µs
Bytes recibidos	320000	640000	1280000	2560000	5120000
Tasa de bits medio	639,6 Kb/s	1136,9 Kb/s	2154,74 Kb/s	4196,01 Kb/s	8279,8 Kb/s
Paquetes perdidos	0%	0%	0%	0%	0%

Tabla 6. Datos obtenidos con el componente ITGDec sobre el *log* de recepción

4.2.2 Envío de tráfico de relleno con limitación de ancho de banda

Para poder ver mejor el comportamiento, simulando un entorno más real, en la siguiente prueba se pondrá un límite de ancho de banda, provocando pérdidas y observando como afecta a los parámetros medidos. Los datos de los enlaces tras la modificación del ancho de banda serán los siguientes.

Velocidad de enlace	Dispositivo A	Interfaz de dispositivo A	Dispositivo B	Interfaz de dispositivo B
Ilimitada[1]	h21	h21-eth0	s2	s2-eth1
Ilimitada	h22	h22-eth0	s2	s2-eth2
3,5 Mbps	s2	s2-eth3	s1	s1-eth1
3,5 Mpbs	s1	s1-eth2	s3	s3-eth3
Ilimitada	s3	s3-eth1	h31	h31-eth0
Ilimitada	s3	s3-eth2	h32	h32-eth0

Tabla 7. Conectividad enlaces con limitación de ancho de banda en la red SDN

Para comprobar la velocidad entre el host h21 y el host h31 usaremos la herramienta Iperf. Para ello estableceremos un servidor en el host h21 que permanecerá a la escucha del cliente, situado en el host h31. Se medirá la velocidad utilizando el protocolo UDP, enviando paquetes de 10 Mbps. Tras esta configuración se ejecuta una prueba de medición de ancho de banda con Iperf, dando como resultado un ancho de banda de 3,4 Mbps, tras el envío de 11 paquetes de 10 Mbps.

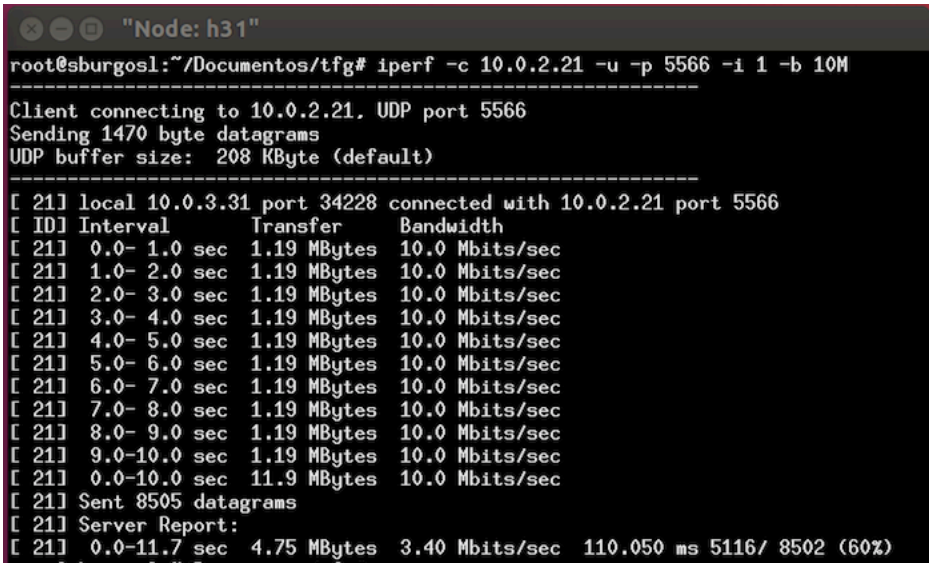


Figura 25. Medición de ancho de banda con Iperf

Volviendo a realizar el envío de tráfico, con los mismos datos de la Tabla 5 obtendremos los siguientes resultados:

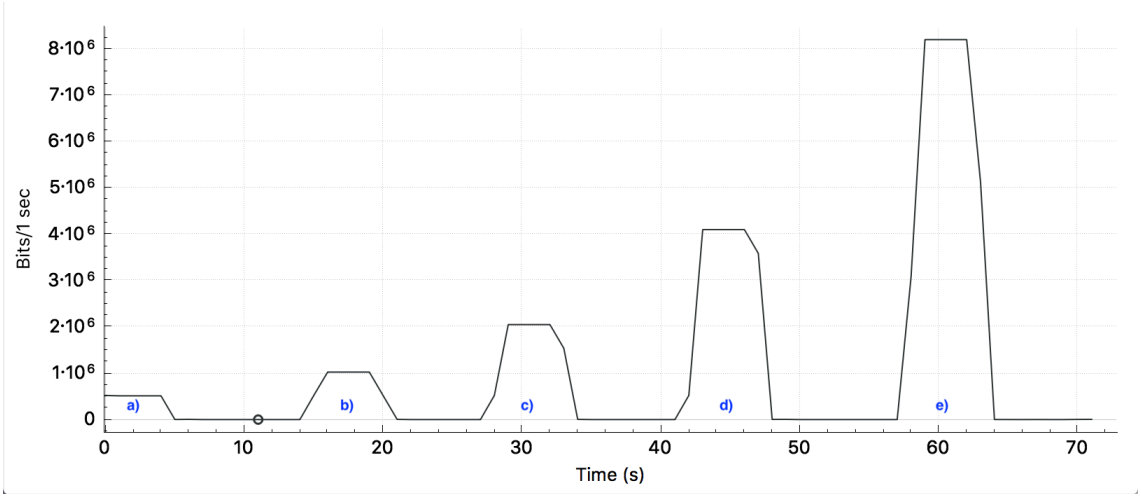
	Prueba a)	Prueba b)	Prueba c)	Prueba d)	Prueba e)
Paquetes enviados	5	10	20	40	79
Retardo mínimo	147 µs	126 µs	112 µs	114 µs	86,52 ms
Retardo máximo	145,8 ms	146,48 ms	241 µs	819,33 ms	6,61 s
Media del retardo	29,3 ms	14,82 ms	142 µs	408,6 ms	3,31 s
σ del retardo	58,26 ms	43,88 ms	30 µs	243,35 ms	1,92 s
Media del *Jitter*	36,43 ms	16,28 ms	34 µs	21 ms	83,37 ms
Bytes recibidos	320000	640000	1280000	2560000	5120000
Tasa de bits medio	663,42 Kb/s	1174,56 Kb/s	2153,05 Kb/s	3591,88 Kb/s	3589,42 Kb/s
Paquetes perdidos	0%	0%	0%	0%	0%

Tabla 8. Datos obtenidos con el componente ITGDec sobre el *log* de recepción, con limitación de ancho de banda.

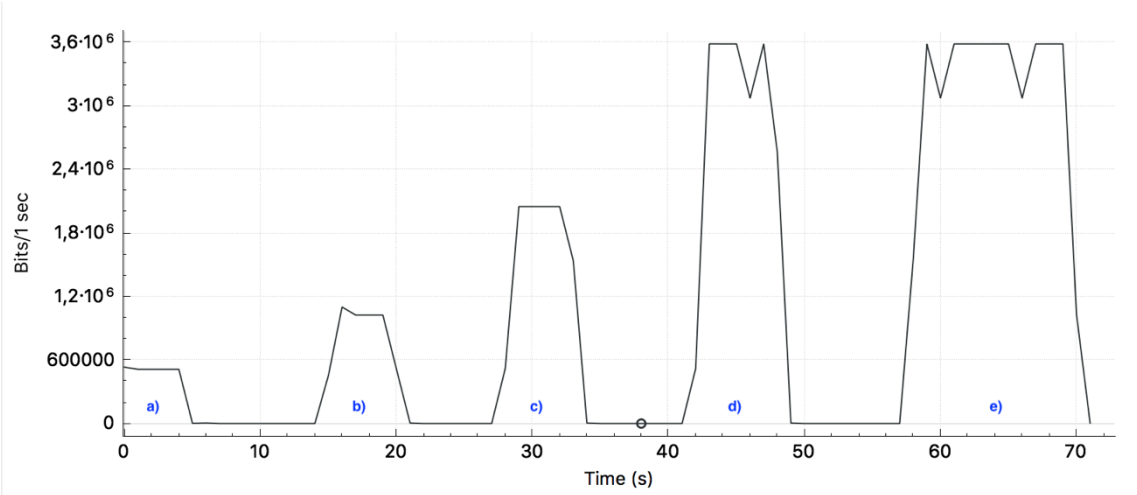
Comparando los resultados obtenidos en las pruebas de envío de tráfico, con limitación y sin limitación, la principal diferencia se encuentra en los tiempos de retardos y *jitter* para las pruebas d) y e), siendo la media, del retardo sin limitar el ancho de banda para la prueba e), de 196 µs y para la red limitada con ancho de banda 3,31 µs para la misma prueba. También se puede ver una gran diferencia en la tasa de bits medio, viendo en la Tabla 6 una tasa de bits similar al ancho de banda recibido, mientras que en la Tabla 8 la tasa de bits no supera los 3,6 Mbps debido a la limitación.

Es posible apreciar que, en ninguna de las pruebas, tanto con limitación como sin limitación de ancho de banda, existen pérdidas. En el caso de las pruebas d) y e), cuando existe limitación, repercute en el tiempo total de cada prueba aumenta desde los 5 segundos de envío a los 11

segundos. Esto es posible visualizarlo viendo la gráfica proporcionada por Wireshark del número de bits enviados y recibidos, en la cual se aprecia en la recepción claramente la limitación de ancho de banda y la duración extendida de dichas pruebas.



Gráfica 3. Bits/s emitidos desde h21 de las pruebas con D-ITG



Gráfica 4. Bits/s recibidos en h31 de las pruebas con D-ITG

4.3 Envío de tráfico multimedia

Como se ha visto, el tráfico multimedia supone un gran problema en las redes actualmente, por ello incluiremos en las pruebas realizadas tráfico multimedia.

El vídeo que se usará para realizar todas las pruebas es el corto animado “Big Buck Bunny” [25] de la fundación Blender. Las características del vídeo se pueden obtener a partir de la información del programa VLC mostrada en la siguiente Figura.

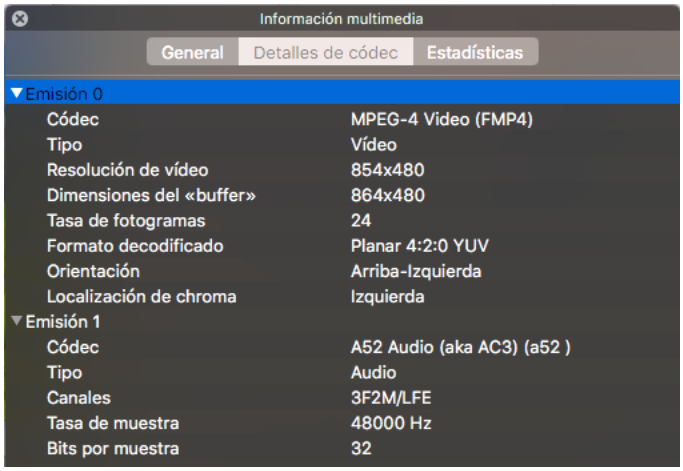


Figura 26. Detalles del contenido multimedia

Utilizando la topología creada, con la configuración de la Tabla 3 y la Tabla 4, mandaremos el vídeo desde el host h21 hasta el host h31. Para ello, a través de VLC estableceremos el emisor en h21 y como cliente en h31. Para la configuración es necesario establecer la IP y puerto del host h31 en el emisor y elegir la encapsulación MPEG2-TS/RTP/UDP. En el receptor únicamente deberemos conectarnos a la emisión al puerto que hayamos especificado.

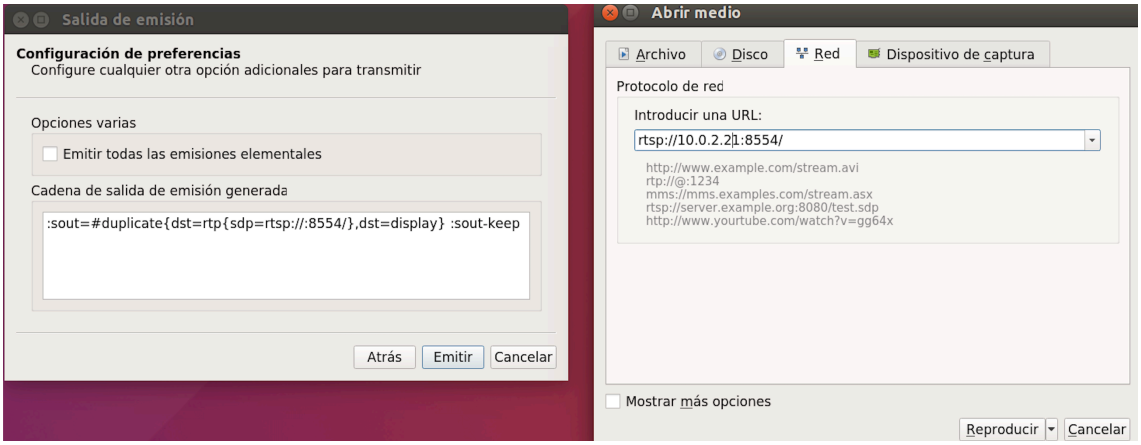
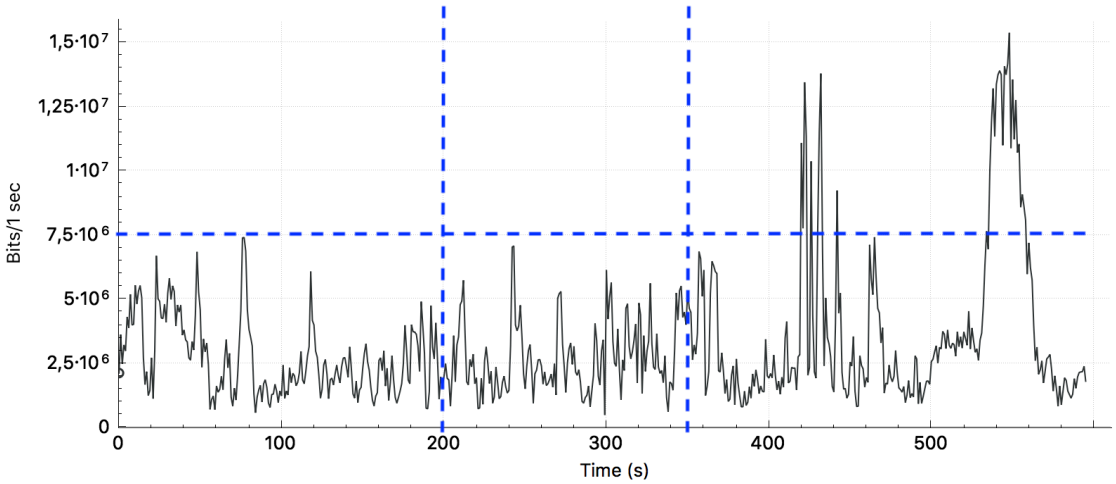


Figura 27. Configuración VLC para el envío y visualización del contenido. Derecha: Emisor. Izquierda: Receptor.

Para obtener la tasa de bits del contenido multimedia, se utilizará Wireshark. VLC proporciona información acerca de la tasa de bits de entrada y de emisión, pero esta información no es válida si se quiere saber el máximo. Para ello a través de las estadísticas se obtendrá la tasa de bits de la recepción del contenido multimedia.



Gráfica 5. Tasa de bits del contenido multimedia

Como se observa el la Gráfica 5, la tasa de bits recibidos no supera en ningún momento los 20 Mbps, pero si obviamos la parte final del vídeo y nos quedamos con el intervalo de tiempo 200-350 segundos, marcado por las líneas azules verticales, la tasa de bits no supera los 7,5 Mbps. Para realizar las siguientes pruebas se usará el mismo contenido multimedia recortado entre el intervalo temporal 200-350 segundos y una configuración de enlaces como la mostrada en la siguiente tabla.

Velocidad de enlace	Dispositivo A	Interfaz de dispositivo A	Dispositivo B	Interfaz de dispositivo B
1 Gbps	h21	h21-eth0	s2	s2-eth1
1 Gbps	h22	h22-eth0	s2	s2-eth2
7,5 Mbps	s2	s2-eth3	s1	s1-eth1
7,5 Mpbs	s1	s1-eth2	s3	s3-eth3
1 Gbps	s3	s3-eth1	h31	h31-eth0
1 Gbps	s3	s3-eth2	h32	h32-eth0

Tabla 9. Conectividad enlaces con limitación de ancho de banda en la red SDN para la transmisión de contenido multimedia y tráfico de relleno

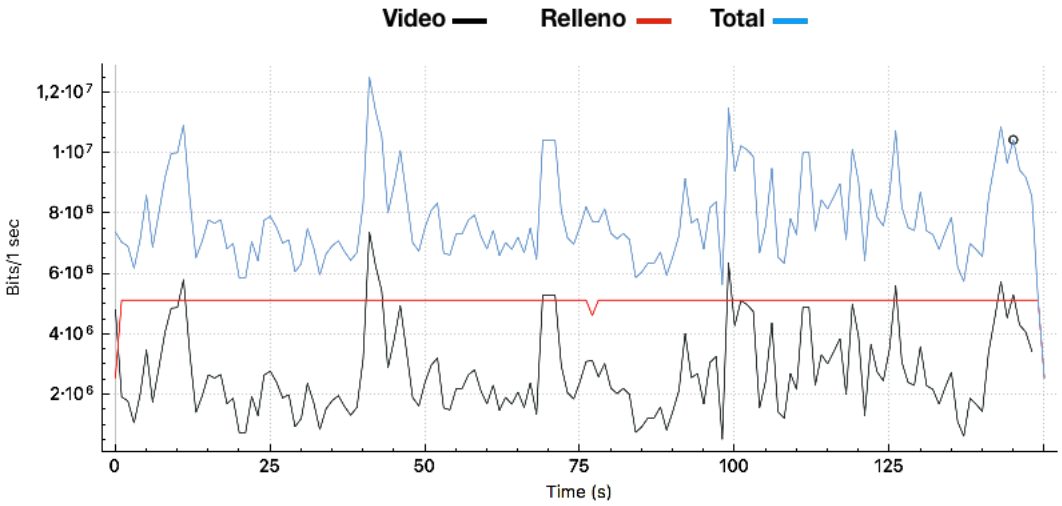
Tras emitir el video recortado, con la nueva configuración en los enlaces, se comprueba que no existen pérdidas y que se obtiene buena visualización.

4.3.1 Envío del contenido multimedia junto con tráfico de relleno

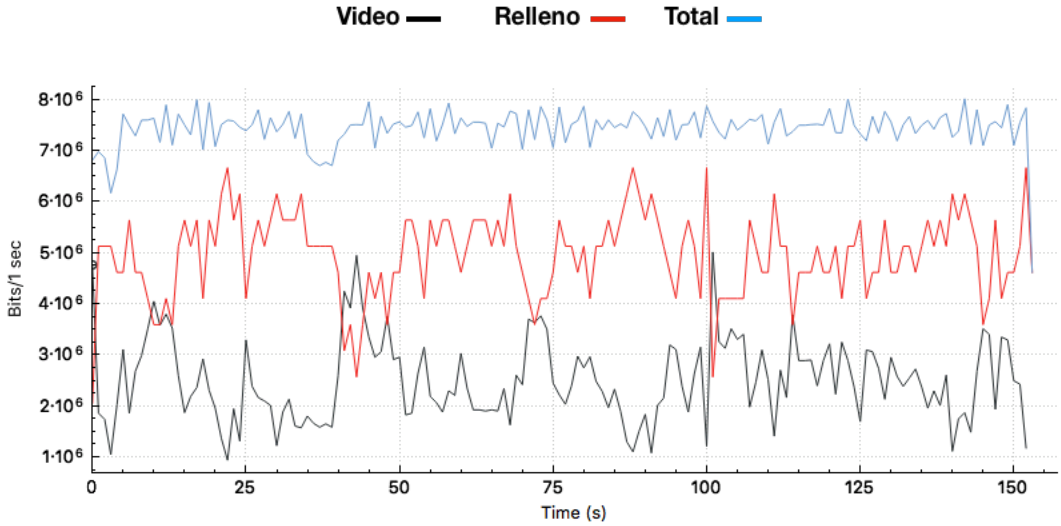
Para ver como afecta la congestión al contenido multimedia, se emitirá tráfico de relleno desde el host h22 hacia el host h32 con la configuración mostrada en la Tabla 10. Como para alcanzar su destino usarán los mismos enlaces entre s1 y s3, en los momentos en los que la tasa de bits enviados supere la limitación se producirán pérdidas o retardos que afectarán en mayor medida al contenido multimedia, debido a que es en tiempo real.

IP Origen	IP Destino	Tamaño paquete (bytes)	Paquetes / Segundo	Duración (ms)	Ancho de banda / Segundo (Mbps)
10.0.2.22	10.0.3.32	64000	10	150000	5

Tabla 10. Configuración tráfico de relleno para el envío junto a contenido multimedia



Gráfica 6. Tasa de bits enviados del contenido multimedia más el tráfico de relleno



Gráfica 7. Tasa de bits recibidos del contenido multimedia más el tráfico de relleno

Tipo de tráfico	Paquetes Enviados	Paquetes Recibidos	Paquetes Perdidos	Porcentaje de pérdidas
Multimedia	37094	34844	2250	6,07 %
Relleno	1499	1490	9	0,6 %

Tabla 11. Estadísticas de los paquetes en la transmisión del contenido multimedia y tráfico de relleno

Como se puede ver en las anteriores gráficas, en la emisión se superan los 7,5 Mbps por la suma del tráfico de relleno y el tráfico multimedia, mientras que en la recepción se establece una media de 7,5 Mbps por el límite de ancho de banda establecido en los enlaces.

Al igual que ha sucedido con las pruebas de envío de tráfico con ancho de banda limitado, se puede observar mediante los resultados obtenidos con D-ITG un retardo y una tasa de bits media inferior a lo esperado.

Media del retardo	σ del retardo	Media del *jitter*	Bytes recibidos	Tasa de bits media
2,17 s	1,15 s	35,67 ms	95360000	4987,17 Kb/s

Tabla 12. Datos obtenidos con el componente ITGDec sobre el *log* de recepción, con limitación de ancho de banda y envío de tráfico multimedia

Al producirse congestión se observa un mayor impacto en el contenido multimedia, produciéndose pérdidas y empeorando la calidad de experiencia con cortes en audio y vídeo, y una mala visualización. En cuanto al tráfico de relleno se pueden observar unas mínimas pérdidas, afectando en mayor lugar al retardo.

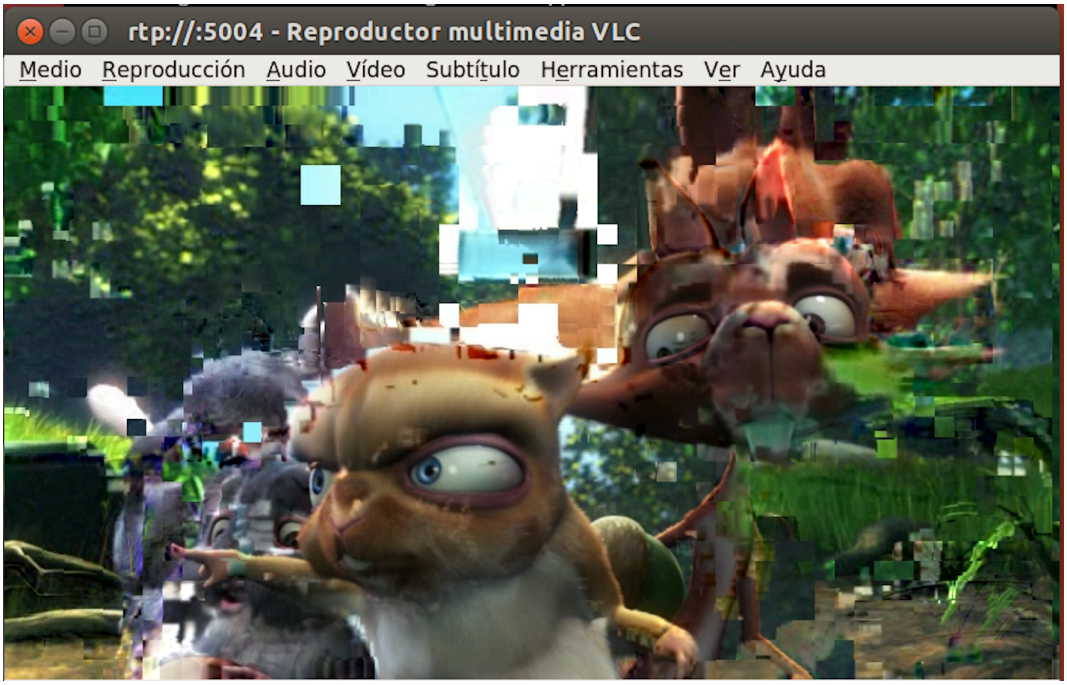
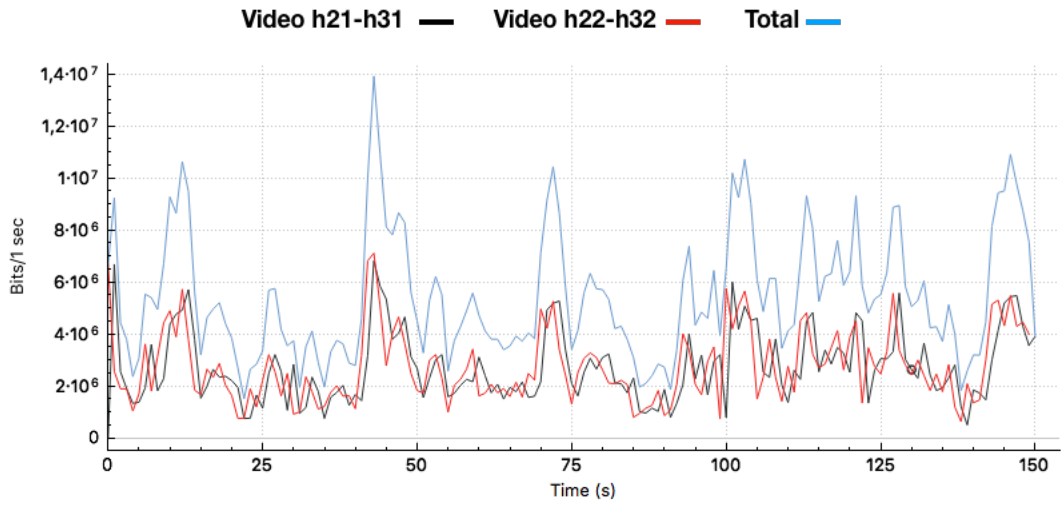


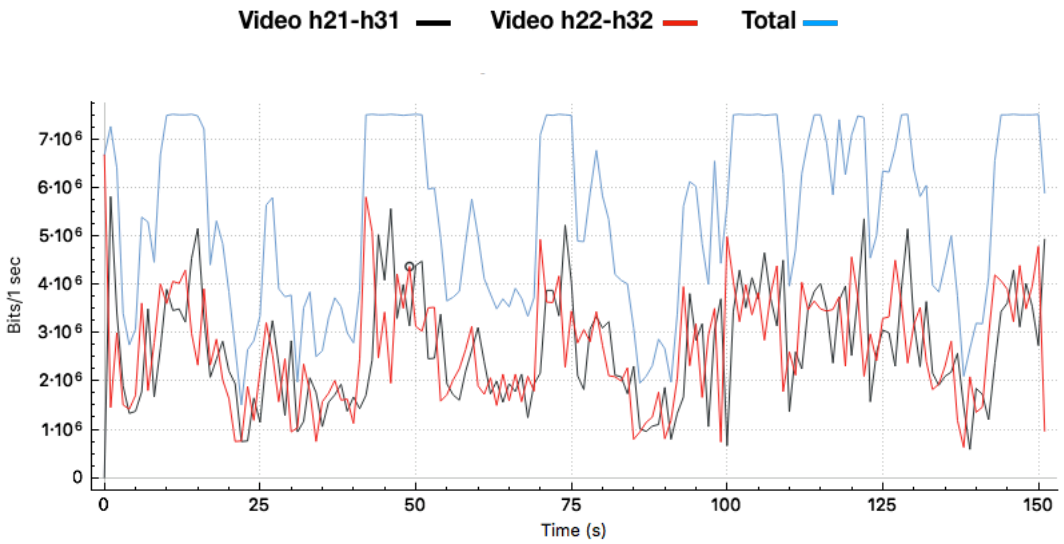
Figura 28. Errores producidos en la visualización del contenido multimedia con congestión en la red

4.3.2 Envío de varios contenidos multimedia

En la siguiente prueba se emitirán el mismo contenido multimedia por duplicado, un flujo entre el host h21 y h31, y otro flujo entre los hosts h22 y h32.



Gráfica 8. Tasa de bits enviados de los contenidos multimedia



Gráfica 9. Tasa de bits recibidos de los contenidos multimedia

Emisor	Receptor	Paquetes Enviados	Paquetes Recibidos	Paquetes Perdidos	Porcentaje de pérdidas
h21	h31	37462	36914	548	1,46 %
h22	h32	37554	36870	684	1,81 %

Tabla 13. Estadísticas de los paquetes en la transmisión de los contenidos multimedia

Tras la prueba se puede comprobar que la tasa de pérdidas se ha reducido respecto al envío de tráfico junto con el contenido multimedia. Mientras que el tráfico tenía una tasa bits de envío constante los contenidos multimedia tienen grandes variaciones, haciendo que únicamente en ciertos puntos se alcance la limitación. El problema es que las pérdidas afectan a ambos flujos, produciendo en ambos clientes que disminuya la calidad de experiencia producida por errores o retardos.

Como solución en una red, se podría aumentar el ancho de banda de los enlaces y se transmitiría sin pérdidas, como se ha visto en las primeras pruebas. Pero esta solución sería temporal, ya que ancho de banda requerido para la transmisión de contenidos multimedia aumenta año tras año y habría que estar cambiando los enlaces con el aumento de hosts, con el aumento de la calidad o tamaño de los contenidos. Por ello, lo más habitual en una red será de disponer de múltiples caminos, con diferente número de saltos entre origen y destino.

4.3.3 Creación topología con varios caminos

Para poder ver la función que tendría un controlador sobre una red con múltiples caminos crearemos la siguiente topología de red para poder realizar las mismas pruebas que hasta ahora y viendo como se comporta el tráfico transmitido.

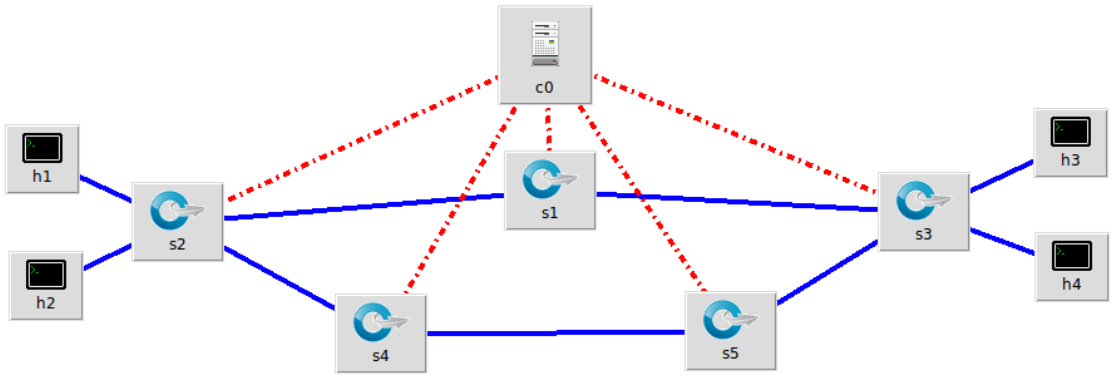


Figura 29. Esquema de la nueva topología con dos caminos

Para la creación de esta topología con Mininet usaremos la siguiente configuración.

Tipo de dispositivo	Nombre	IP	Mac	Interfaces
Host	h21	10.0.2.21/16	00:00:00:00:00:21	h21-eth0
Host	h22	10.0.2.22/16	00:00:00:00:00:22	h22-eth0
Host	h31	10.0.3.31/16	00:00:00:00:00:31	h31-eth0
Host	h32	10.0.3.32/16	00:00:00:00:00:32	h32-eth0
Switch OVS	s1			s1-eth1; s1-eth2
Switch OVS	s2			s2-eth1; s2-eth2; s2-eth3; s2-eth4
Switch OVS	s3			s3-eth1; s3-eth2; s3-eth3; s3-eth4
Switch OVS	s4			s4-eth1; s4-eth2
Switch OVS	s5			s5-eth1; s5-eth2

Tabla 14. Configuración dispositivos de la nueva topología de red SDN

Los enlaces estarán conectados según a la información mostrada en la Tabla 15.

Velocidad de enlace	Dispositivo A	Interfaz de dispositivo A	Dispositivo B	Interfaz de dispositivo B
1 Gbps	h21	h21-eth0	s2	s2-eth1
1 Gbps	h22	h22-eth0	s2	s2-eth2
7,5 Mbps	s2	s2-eth3	s1	s1-eth1
7,5 Mpbs	s1	s1-eth2	s3	s3-eth3
7,5 Mbps	s2	s2-eth4	s4	s4-eth1
7,5 Mbps	s4	s4-eth2	s5	s5-eth1
7,5 Mbps	s5	s5-eth2	s3	s3-eth4
1 Gbps	s3	s3-eth1	h31	h31-eth0
1 Gbps	s3	s3-eth2	h32	h32-eth0

Tabla 15. Conectividad enlaces de la nueva topología de red SDN

En esta topología podríamos programar el controlador para que estableciera el encaminamiento de forma similar al protocolo Open Shortest Path First (OSPF), que erigirá el camino con menor coste, o similiar al Protocolo de Información de Encaminamiento (RIP), que eligirá el camino con menor número de saltos. Para esta topología tanto el menor coste como el menor número de saltos es el camino superior, definido por los siguientes dispositivos de forma bidireccional:

```
h21 - s2 - s1 - s3 - h31

h22 - s2 - s1 - s3 - h32
```

En este caso obtendríamos el mismo camino que con el que se han realizado las pruebas anteriores, obteniendo los mismos resultados.

Un controlador no es un enrutador, por lo que para esta topología se programaría para actuar dependiendo a la información obtenida por los conmutadores, a parte de simular el comportamiento de algún protocolo de enrutamiento. Se tendría una red capaz de variar sus decisiones según los parámetros establecidos.

En este caso se introducirán, a través del controlador, las órdenes necesarias para que se transmitan los paquetes siguiendo la ruta superior. En el momento en el que se observe una congestión se variará para que tan solo uno de los flujos modifique su camino, cambiando al inferior.

Para verificar el buen funcionamiento de la red, limitaciones y órdenes recibidas por el controlador, se realizará una prueba de medición de ancho de banda con Iperf. De esta forma se comprobará, con las tablas de flujo en los conmutadores en las que la comunicación entre los hosts h21 y h31 es por el camino superior, a través del conmutador s1, y la comunicación entre h22 y h32 es por el camino inferior, a través de los conmutadores s4 y s5.

En las pruebas realizadas con Iperf se ha obtenido una tasa media de 7,29 Mbps, tanto por el camino superior como por el inferior.

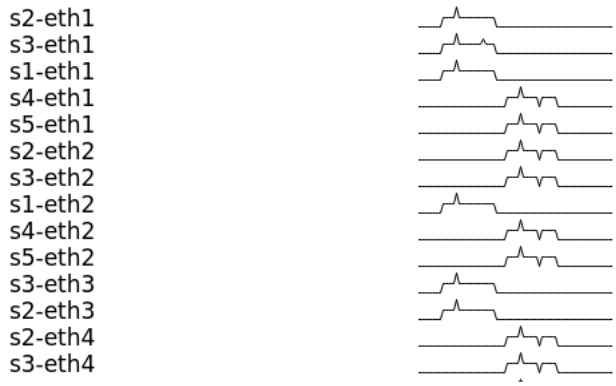


Figura 30. Interfaces de los conmutadores de la red SDN con dos caminos, y el tráfico detectado en ellas durante las pruebas de Iperf.

En la figura anterior se puede comprobar el correcto funcionamiento de la red tras haber introducido en el controlador las órdenes para separar los flujos entre los hosts h21 - h31 y h22 - h32. En la cual se puede observar un inicio en el que se usan los conmutadores:

```
s2 - s1 - s3
```

Y una segunda parte en la que se puede observar el uso de los siguientes conmutadores:

```
s2 - s4 - s5 - s3
```

4.3.4 Envío del contenido multimedia y tráfico de relleno

Una vez comprobado el buen funcionamiento y las limitaciones se reestablecerá la red a utilizar únicamente el camino superior y se procederá a realizar la primera prueba, enviando tráfico de relleno y el contenido multimedia recortado, de acuerdo a las siguientes configuraciones.

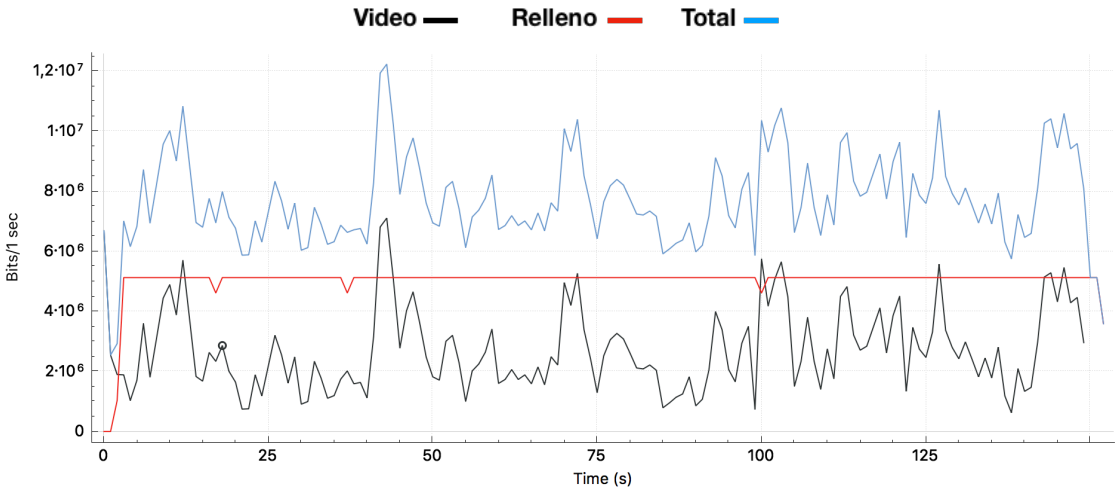
Tipo de contenido	Origen	Destino
Multimedia	h21 - 10.0.2.21	h31 - 10.0.3.31
Relleno	h22 - 10.0.2.22	h32 - 10.0.3.32

Tabla 16. Configuración del tráfico multimedia y tráfico de relleno, para su envío y recepción en una topología con dos caminos

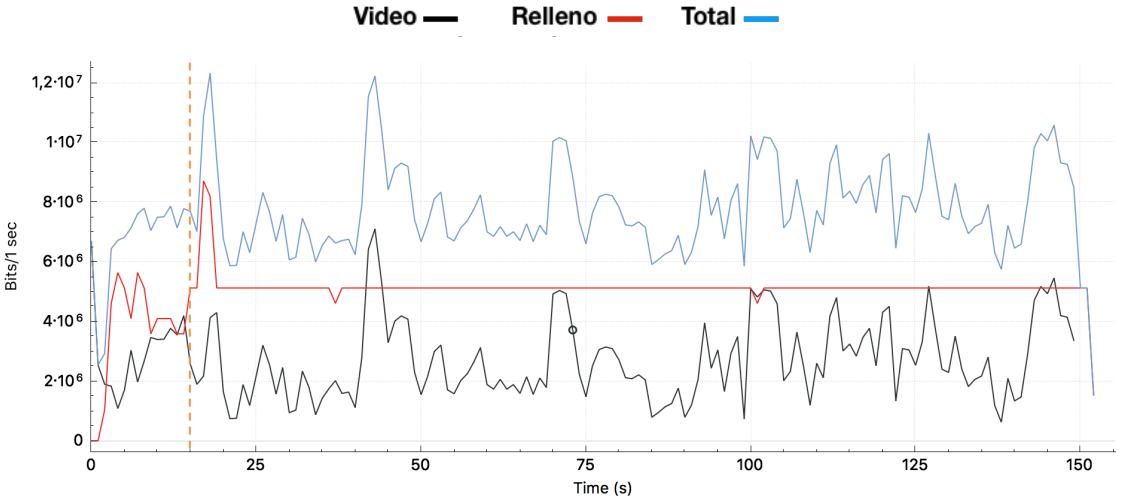
Tamaño paquete (bytes)	Paquetes / Segundo	Duración (ms)	Ancho de banda / Segundo (Mbps)
64000	10	150000	5

Tabla 17. Configuración tráfico de relleno para el envío junto a contenido multimedia en una topología con dos caminos

Tras iniciar la prueba, se ha esperado a detectar congestión en la red para, a través del controlador, cambiar el camino del tráfico de relleno del superior al inferior, manteniéndose el contenido multimedia por el camino superior. Esta detección se ha producido en el segundo 15, donde en la Gráfica 11, a través de una línea discontinua, está marcado dicho momento.



Gráfica 10. Tasa de bits enviados del contenido multimedia y el tráfico de relleno



Gráfica 11. Tasa de bits recibidos del contenido multimedia y el tráfico de relleno

En el momento en el que se ha producido el cambio de las tablas de flujo se puede observar como la tasa de bits de recepción se vuelve similar a la de envío. Antes de ello se produce un aumento en la tasa de bits recibidos del tráfico de relleno, correspondientes a los paquetes que se encontraban en cola por la congestión.

En cuanto a las pérdidas producidas son casi despreciables, obteniendo una visualización del contenido multimedia sin afectar a la calidad de experiencia y un audio reproducido sin cortes.

Tipo de contenido	Emisor	Receptor	Paquetes Enviados	Paquetes Recibidos	Paquetes Perdidos	Porcentaje de pérdidas
Multimedia	h21	h31	37461	37452	9	0,02%
Relleno	h22	h32	1497	1492	5	0,33%

Tabla 18. Estadísticas de los paquetes en la transmisión del contenido multimedia y tráfico de relleno, en una topología de red con dos caminos

Si se observa con detalle los resultados obtenidos, con el componente ITGDec, del tráfico de relleno, se puede ver como a pesar de las pérdidas y del retardo sufrido al inicio de la prueba, debido a la limitación de ancho de banda, la tasa de bits media se corresponde al ancho de banda utilizado por segundo. También se pueden observar valores inferiores de *jitter* y retardo respecto a las pruebas anteriores con limitación.

Media del retardo	σ del retardo	Media del *jitter*	Bytes recibidos	Tasa de bits media
65 ms	281,49 ms	24,66 ms	95744000	5110,64 Kb/s

Tabla 19. Datos obtenidos con el componente ITGDec sobre el *log* de recepción, en la topología de red con dos caminos y actuación del controlador.

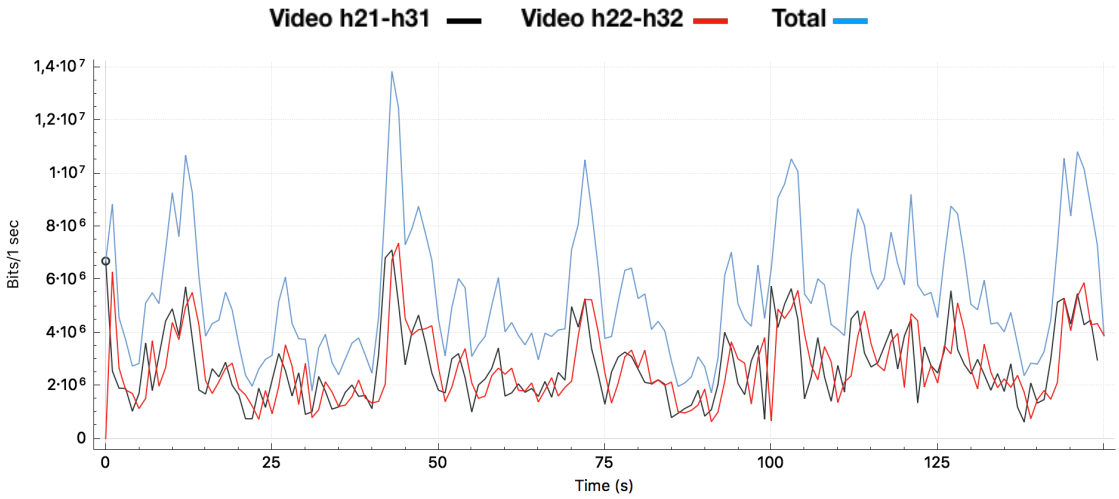
4.3.5 Envío de varios contenidos multimedia

Se realiza la misma prueba, pero cambiando el tráfico de relleno por el mismo contenido multimedia para observar las estadísticas de ambos contenidos multimedia. Para ello se establecerá la siguiente configuración con VLC.

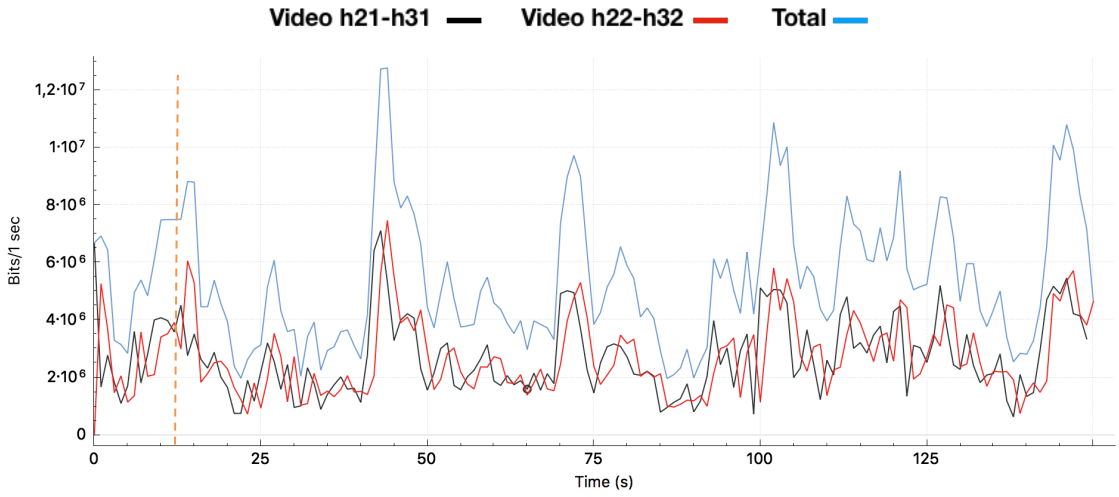
Tipo de contenido	Origen	Destino	Puerto
Multimedia	h21 - 10.0.2.21	h31 - 10.0.3.31	5004
Multimedia	h22 - 10.0.2.22	h32 - 10.0.3.32	5014

Tabla 20. Configuración del tráfico multimedia para su envío y recepción en una topología con dos caminos

Al igual que en la prueba anterior, se ha esperado a detectar congestión en la red para variar el camino del contenido multimedia entre los hosts h22 y h32 para que pase del superior al inferior, manteniendo el camino superior para el flujo entre los hosts h21 y h31. Esta congestión se ha producido en el segundo 13, el cual está marcado a través de una línea discontinua en la Gráfica 13.



Gráfica 12. Tasa de bits enviados de los contenidos multimedia con actuación del controlador



Gráfica 13. Tasa de bits recibidos de los contenidos multimedia con actuación del controlador

Antes de ejecutar el cambio en las tablas de flujo de los conmutadores se llega dos veces al límite del ancho de banda, en la segunda vez, más prolongada, es cuando se ejecuta el cambio y se normaliza la tasa de bits recibidas, siendo similar a la tasa de bits enviados.

Las pérdidas producidas son muy inferiores a las mismas producidas en el envío de ambos flujos sobre el mismo camino. Con ello se obtiene una visualización del contenido multimedia sin afectar a la calidad de experiencia y un audio reproducido sin cortes.

Tipo de contenido	Emisor	Receptor	Paquetes Enviados	Paquetes Recibidos	Paquetes Perdidos	Porcentaje de pérdidas
Multimedia	h21	h31	37462	37456	6	0,02%
Relleno	h22	h32	37464	37420	44	0,12%

Tabla 21. Estadísticas de los paquetes en la transmisión de los contenidos multimedia, en una topología de red con dos caminos

Capturando un instante de la visualización de los contenidos multimedia, en ambos hosts, durante un tramo en el que anteriormente existían pérdidas, empeorando la calidad de experiencia en ambos receptores, se puede observar su correcta visualización.

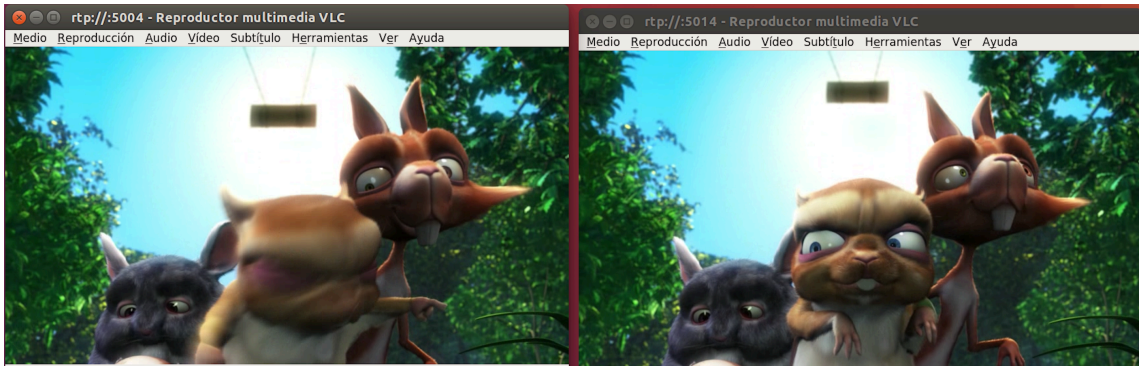


Figura 31. Visualización de la recepción de ambos flujos multimedia en una red SDN

Capítulo 5. Conclusiones y líneas de trabajo futuras

5.1 Conclusiones

Durante la realización de este trabajo se ha tratado con detalle la forma de trabajo de las redes SDN, así como sus ventajas con respecto a las redes tradicionales. Tras la explicación de estas redes se han procedido a realizar pruebas con diversas herramientas, previamente analizadas, para comprobar el funcionamiento de una red SDN durante el envío de tráfico de relleno y tráfico multimedia.

Con ello se puede decir que se han cumplido los objetivos marcados para este trabajo, remarcando los siguientes puntos que han reforzado el objetivo de enviar tráfico por una red SDN con Mininet.

- Se ha estudiado el emulador Mininet. En él se han realizado pruebas sobre topologías predefinidas, modificando parámetros y analizando los cambios producidos. También se han desarrollado topologías personalizadas completamente, lo que ha permitido tener un control completo sobre los dispositivos y la topología de la red, pudiendo variar cada parámetro para conseguir realizar las pruebas propuestas.

- Se ha estudiado la herramienta Miniedit, sirviendo de gran ayuda mediante el diseño gráfico de una topología y su exportación a Python para poder ejecutarse en Mininet.

- Se han estudiado herramientas de generación de tráfico como Iperf o D-ITG. La primera más centrada en realizar mediciones de ancho de banda entre dos hosts y la segunda más centrada en la generación y personalización del tráfico que se desea generar, pero también con la capacidad de realizar mediciones proporcionando grandes cantidades de parámetros.

- Se ha podido visualizar y analizar el protocolo RTP a través del envío del contenido multimedia.

- Se ha comprobado la importancia del controlador en la red, estableciendo tablas de flujo manualmente y obteniendo una respuesta por la red que permitía saber el buen funcionamiento de las acciones introducidas.

- Se han visto los efectos que supone para el tráfico, o el consumo de contenidos multimedia, la congestión en la red.

Con la realización de pruebas sencillas se ha podido ver una gran utilidad en el uso del controlador. Aunque para la solución llevada a cabo por el controlador durante estas pruebas puede ser realizada por otros dispositivos, como un balanceador de carga, que también se puede implementar por algunos protocolos como RIP u OSPF, sin ser necesario establecer una red SDN, este debería ser configurado previamente y no actuaría de una manera inteligente como el controlador. Esto es tan solo un ejemplo de las capacidades que se tienen. Un controlador puede obtener información sobre toda la red y actuar en función de múltiples parámetros, pudiendo aplicar todas las medidas oportunas para ello.

En cuanto al tráfico multimedia, el principal efecto que se puede percibir por los usuarios en su consumo es el retardo y las pérdidas, ambos conceptos analizados en las pruebas realizadas, por lo que es importante disponer de una topología controlada para evitar congestiones que

pueden ser evitadas, sin tener que aumentar el ancho de banda de los enlaces cada vez que se requiera más, aunque en ocasiones puede ser necesario si no existe otra opción.

5.2 Líneas de trabajo futuras

Como continuación a este trabajo se podría desarrollador un controlador que actúe de forma autónoma en función de los datos recibidos por los conmutadores.

Al saber ya las acciones que se deben ejecutar, se necesitará estudiar la creación de un software que pueda ser ejecutado en el controlador, analizando los datos entrantes, haciendo que genere mensajes para obtener información acerca de la congestión en los conmutadores y actuando en función de ella.

Capítulo 6. Bibliografía

[1] Ronda Hauben (2004). "The Internet: On its International Origins and Collaborative Vision", [En línea]. Available: http://www.ais.org/~jrh/acn/ACn12-2.a03.txt.

[2] CNMC DATA, 2017. [En línea]. Available: http://data.cnmc.es/datagraph/jsp/inf_anual.jsp.

[3] Cisco Visual Networking Index (VNI) Forecast and Service Adoption, 2017. [En línea]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf.

[4] ITU-T E.800, [En línea]. Available: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-E.800-198811-S!!PDF-E&type=items.

[5] ITU-T FG IPTV. Definition of Quality of Experience (QoE), [En línea]. Available: https://www.itu.int/md/T05-FG.IPTV-IL-0050/es.

[6] NTT Communications. CDN, [En línea]. Available: https://www.ntt.com/en/services/network/cdn.html.

[7] Open Networking Foundation (ONF), [En línea]. Available: https://www.opennetworking.org/mission/.

[8] Software-Defined Networking: A Perspective from within a Service Provider Environment, [En línea]. Available: https://tools.ietf.org/html/rfc7149.

[9] Understanding the SDN Architecture – SDN Control Plane & SDN Data Plane, [En línea]. Available: https://www.sdxcentral.com/sdn/definitions/inside-sdn-architecture/.

[10] Open Networking Foundation. SDN Architecture, [En línea]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_SDN_ARCH_1.0_06062014.pdf.

[11] Wei Zhou, Li Li, Min Luo, Wu Chou , «REST API Design Patterns for SDN Northbound API».

[12] Open Networking Foundation (ONF), [En línea]. Available: https://www.opennetworking.org/news-and-events/press-releases/open-networking-foundation-introduces-northbound-interface-working-group/.

[13] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker y J. Turner, «OpenFlow: Enabling innovation in campus networks,» *ACM SIGCOMM Computer Communications Review,* Apr. 2008.

[14] Open Networking Foundation (ONF). OpenFlow Switch Specification, [En línea]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf.

[15] EstiNet Technologies Inc., [En línea]. Available: http://www.estinet.com/.

[16] NS-3. [En línea]. Available: https://www.nsnam.org.

[17] Mininet, [En línea]. Available: http://mininet.org/download/.

[18] Introduction to Mininet, [En línea]. Available: https://github.com/mininet/mininet/wiki/Introduction-to-Mininet#what.

[19] Videolan (VLC), [En línea]. Available: https://www.videolan.org/vlc/index.ex.html.

[20] Wireshark. [En línea]. Available: https://www.wireshark.org/about.html#authors.

[21] Python, [En línea]. Available: https://wiki.python.org/moin/FrontPage.

[22] GNU Operating System, [En línea]. Available: https://www.gnu.org/software/bash/manual/bash.html#What-is-Bash_003f.

[23] Iperf, [En línea]. Available: https://iperf.fr.

[24] D-ITG, [En línea]. Available: http://www.grid.unina.it/software/ITG/manual/.

[25] Big Buck Bunny, [En línea]. Available: https://peach.blender.org.