Universidad Politécnica de Madrid
Escuela Técnica Superior de Ingenieros de Telecomunicación


# DESARROLLO DE APLICACIONES DE INGENIERÍA DE TRÁFICO EN REDES WAN BASADAS EN SOFTWARE

**TRABAJO FIN DE MÁSTER**

**Andrés Jorge Muracciole Vázquez**

2020

Universidad Politécnica de Madrid
Escuela Técnica Superior de Ingenieros de Telecomunicación

**Máster Universitario en**
**Ingeniería de Redes y Servicios Telemáticos**

**TRABAJO FIN DE MÁSTER**

# DESARROLLO DE APLICACIONES DE INGENIERÍA DE TRÁFICO EN REDES WAN BASADAS EN SOFTWARE

Autor
**Andrés Jorge Muracciole Vázquez**

Director
**Carlos Mariano Lentisco Sánchez**

Departamento de Ingeniería de Sistemas Telemáticos

2020

## Resumen

Vivimos en una época donde las tecnologías se encuentran en constante proceso evolutivo. Cada día surgen nuevas ideas, aplicaciones y servicios que satisfacen las necesidades de los usuarios, buscando alcanzar mejores estándares de calidad y haciendo mas instantánea y real la experiencia. Las plataformas migran hacia la nube, el tráfico es cada vez mayor y con ello las exigencias también. Sin embargo las redes por las cuales se transportan los datos parecieran como si no se hubieran enterado de estos avances ya que no han sufrido mayores cambios desde sus orígenes.

Como forma de acompañar dicho crecimiento y previendo la demanda a futuro surge la necesidad de aplicar métodos por los cuales se pueda obtener mayores beneficios de las redes. A raíz de ello es que nacen las *redes definidas por software*, las cuales mediante técnicas de computación permiten facilitar la implementación de servicios de una manera mucho mas dinámica, escalable y centralizada. A su vez, se logra desacoplar el plano de datos del de control, con lo cual la gestión y administración de la red se ve ampliamente beneficiada ya que admite nuevas técnicas como son las diversas soluciones de ingenierías de tráfico, de acuerdo a las necesidades propias.

Emplear métodos para optimizar el tratamiento de tráfico en las redes WAN no es algo nuevo. Las soluciones MPLS son sin duda la primordial implementación en las redes tradicionales si se quiere ofrecer una calidad de servicio; sin embargo estas suelen ser costosas y poco gestionables, lo cual trae consigo una gran limitante. La adaptación del concepto SDN a las redes WAN abren un nuevo paradigma de cómo realizar estos procesos los cuales hasta el momento eran impensados. Optimización de enlaces, balanceos o decisiones de enrutamiento en tiempo real a partir del estado de la red es posible con SD-WAN.

Este trabajo logra plantear implementar una solución de ingeniería de tráfico que permita aplicar calidad de servicio sobre tráfico WAN multicamino, seleccionando en tiempo real la ruta óptimo que deben tomar los paquetes. Para ello se diseñó una solución mediante la cual se que monitorea los enlaces WAN en un escenario virtual VNX y por medio de consultas a la API del controlador Ryu permite modificar las tablas de flujo de los conmutadores en tiempo real.

Haciendo uso de esta solución se logró ver una mejora sustancial en cuanto a la escalabilidad y centralización de la administración de la red. Por otro lado se obtuvieron resultados muy positivos del uso de la misma los cuales de plasman en mayores velocidades, menores pérdidas y menores retardos.

## Abstract

# Índice general

# Índice de figuras

## Siglas

| Término | Acrónimo |
|---|---|
| Border Gateway Protocol | BGP |
| Centro de Procesamiento de Datos | CPD |
| Constrained Shortest Path First | CSPF |
| Free Range Routing | FRR |
| Intermediate System to Intermediate System protocol | IS-IS |
| Internet Service Provider | ISP |
| Local Area Network | LAN |
| Overlay Management Protocol | OMP |
| Open Shortest Path Firts | OSPF |
| OpenvSwitch | OVS |
| Quality of Service | QoS |
| Routing Information Protocol | RIP |
| Software Define Network | SDN |
| Software Define Wide Area Network | SD-WAN |
| Traffic Engineer | TE |
| Virtualized Network Service | VNS |
| Virtual Networks over Linux | VNX |
| Vector Packet Processor | VPP |
| Wide Area Network | WAN |

# 1 Introducción

Llegó el momento de cambiar los conceptos de redes con los cual hemos crecido y madurado. El mundo se mueve a pasos agigantados y los avances tecnológicos no se quedan atrás. Cada día que pasa hay más dispositivos conectados, más aplicaciones realizando peticiones a servidores, más servicios al usuario y por ende… más tráfico sobre la red. Para poder cubrir y satisfacer las necesidades de los clientes, las redes deben crecer a la misma velocidad o mayor si quieren cubrir la demanda futura. Enlaces mas grandes, pluralidad de caminos, equipos mas potentes, son algunas de las formas que se tiene de lograr este crecimiento. Sin embargo, el concepto de trasfondo de lo que conocemos como red no ha cambiado en años. Los paquetes se conmutan de la misma forma y los protocolos de enrutamiento no varían. Sin embargo la red se vuelve cada vez más compleja, transportando una gran volumen diverso de tráfico, haciendo así que la administración de complejice pero a la vez debido a las prestaciones que ofrece, la vuelve rígida, costosa y poco flexible.

Este es uno de los principales problemas que se comenzó a apreciar en las redes de comunicaciones hace ya varios años. Para realmente acompañar la creciente demanda mencionada anteriormente no solo era necesario mas y mejor equipamiento. Faltaba un pilar primordial que permitiera darle mas flexibilidad y adaptación a las redes. Con esta necesidad y el alcance que se tiene con la virtualización de servicios es que surgen las redes definidas por software SDN.

Consecuencia de este crecimiento es que se afianzan los ya conocidos IaaS, PaaS y SaaS. Con esto, la migración de servicios alojados en la nube (tanto pública como privada) han aumentado notoriamente haciendo que el tráfico local en muchos casos se migre a un tráfico sobre internet. Nuevamente esta migración no podría ser posible de no ser por las ya mencionadas SDN, pero en este caso al tratarse de redes públicas se las conoce como SD-WAN.

Tanto los líderes tecnológicos (Google, Microsoft, etc.) como las empresas telefónicas ven la necesidad y potencial de migrar sus servicios hacia aquí. De esta forma es que surgen diversas propuestas para el tratamiento de datos sobre redes WAN. En algunos casos redes privadas, en otros públicas y muchas veces mixtas. Como sabemos internet es una red best effort, con lo cual no permite aplicar políticas de calidad de servicio. Todos los usuarios compiten con un espacio el canal de comunicaciones y sus servicios se verán afectados por el tráfico se sus vecinos. Sin embargo mediante soluciones de ingeniería de tráfico sobre redes WAN es posible aplicar, de cierta forma, políticas de priorización de tráfico. Balancear la carga, determinar el canal por el cual enviar un determinado tráfico o tomar esas decisiones de forma automatizada en función del estado de la red, son soluciones que permiten mejorar el uso de la red.

Este proyecto de fin de master consiste en la implementación y desarrollo de una solución de ingeniería de tráfico (llamada de aquí en más también TE por sus siglas en inglés "traffic engeneering") la cual busca en resolver y optimizar el tráfico que cursan dos oficinas remotamente ubicadas. Estas cuentan con la particularidad que cuentan con mas de una conexión WAN, donde una de ellas es una red SDN con topología de pez multicamino. Sobre esta es que se hará el diseño de la TE.

La idea principal consiste por un lado en realizar una categorización el tráfico sensible para el cual se requieran mayores prestaciones. Este es el que se desea priorizar respecto a los demás. Para ello se monitorearán los diversos caminos de la red para llegar al destino y se programará una aplicación que permita tomar decisiones en tiempo real de por donde enviar el tráfico sensible sin generar mayores inconvenientes en el otro. Con esto se buscará sacarle el mejor beneficio a la infraestructura a un bajo costo, e forma programática, escalable y rápida.

Una vez implementado se realizarán pruebas comparativas para determinar la utilizad de una solución de este tipo y entender finalmente el potencial que implica esto tanto para el usuario final como para la empresa proveedora del servicio.

Para llevar a cabo esta tesis, se trabajará sobre un escenario virtual implementado por el autor de la misma, la cual estará funcionando sobre VNX. Esta es una plataforma de código abierto implementada por el Departamento de Ingeniería de Telecomunicaciones (DIT) de la Universidad Politécnica e Madrid (UPM), que permite desplegar escenarios virtuales. Por otra parte una solución SDN debe contar con un controlador. En este casó se utilizará Ryu ya que también es de código abierto y permite acoplarle una amplia gama de aplicaciones ya existentes. Una de estas es la llama Traffic Monitor la cual será de gran utilizad a la hora de querer obtener las métricas de los enlaces en tiempo real. Con ella se puede conocer desde la cantidad de paquetes cursados por cada interfaz hasta el ancho de banda consumido en tiempo real. Este ultimo valor será importante cuando se requiera tomar las decisiones de encaminamiento.

Cabe destacar que la ingeniería de tráfico de este trabajo consiste en decidir si es necesario aplicar políticas (modificar tablas de flujo) en función del tráfico a enviar y el estado instantáneo de la red. Para ello se hará un diseño que simplifique las tareas del administrador y que le permita mediante una interfaz gráfica amigable poder realizar configuraciones en la red de manera casi instantánea, que de otra forma tomaría mucho mas tiempo y dedicación. Para que todo esto funcione, fue necesario desarrollar varios scripts e integrarlos para que trabajen conjuntamente pero que a la vez las llamadas a estos sean transparentes para el administrador, el cual solo interactuará desde un panel central.

# 2 Estado del arte

Las redes LAN (Local Area Network) son sin duda las mas utilizadas a nivel empresarial. Allí radican los servicios internos así como también los activos tales como impresoras, registros y servidores de información los cuales en un principio solo interesa que se tenga acceso desde dentro de la corporación. Sin embargo, en muchos casos también es requerida la comunicación desde fuera de dicha red, ya sea para obtener cierta información interna como también para utilizar recursos de la misma. Es aquí donde entra en juego el rol de las redes WAN (Wide Area Network). Estas permiten interconectar redes LAN que en muchos casos están físicamente a cientos o miles kilómetros de distancia y simular como si estuvieran directamente conectadas entre sí. De no existir las redes WAN, las empresas que cuentan con mas de una sucursal deberían tener replicado su equipamiento en cada una de los sitios haciendo así poco escalable el negocio. De esta forma un usuario en Barcelona puede acceder al registro de facturación alojado en las oficinas de Madrid, o viceversa, sin siquiera saber que los paquetes se encuentran cruzando cientos de kilómetros de distancia. Sin embargo, las redes WAN son de acceso público y por ende en ellas viaja información tanto propia como de muchas otras empresas y usuarios haciendo que sea imposible asegurar y estimar la calidad de servicio a la que están expuestos. En definitiva no hay que olvidarse que internet es *Best Effort*, con lo cual no hay forma de priorizar los paquetes. A pesar de que por algún motivo se modifiquen las cabeceras a nivel de red local, una vez que entra a internet todo el tráfico es tratado de igual forma. A raíz de esto es que las empresas suelen contratar a los proveedores de servicios unas conexiones dedicadas para así asegurarse un nivel de QoS aceptable en sus aplicaciones. Ejemplo de estas son las conocidas MPLS (Multiprotocol Label Switching) las cuales se implementan sobre redes WAN y a diferencia de redes IP, el protocolo de encaminamiento es mas ágil y sencillo debido a que se hace mediante el uso de etiquetas y no de direcciones. De este modo una empresa puede tener dos o mas enlaces WAN (muchas veces provistos por diferentes ISP´s) y decidir cuál tipo de tráfico es mas sensible y en función de ello enviarlo por el canal con mayores prestaciones. A pesar de ello, estos servicios suelen ser bastante costosos para las empresas y poco flexibles para el proveedor.

Por otra parte desde hace ya unos años la demanda de recursos y cómputo viene en constante crecimiento al punto de que aplicaciones que antes residían en CPD´s propios, hoy se ejecutan en la nube pública. Esto trae consigo un aumento en el tráfico hacia internet, haciendo que este sea el cuello de botella y genere saturación.

Es a raíz de todo esto que las redes SDN vienen a mejorar estos problemas dando mas flexibilidad de una manera mas ágil y económica. Con este nuevo paradigma un operador puede al cabo de unos pocos minutos levantar un nuevo sitio, realizar configuraciones remotas y establecer criterios de flujo para decidir bajo demanda el camino mas óptimo por el cual enviar los paquetes en función de las prestaciones que se

requieran. A su vez, con la inclusión de la virtualización de funciones de red (VNF) es posible realizar un despliegue de equipamiento de red virtualizado como si fuera físico disminuyendo el tiempo de ejecución y puesta en marcha significativamente respecto al método tradicional. Muchas veces los enlaces WAN son contratados a diferentes empresas; sin embargo, las soluciones SD-WAN son independientes a esto, permitiendo trabajar con ellas sin importar si el proveedor es o no el mismo que brinda la solución SD-WAN. En definitiva, permite unificar las redes WAN y tratarla como si fuera "una sola" bajo demanda y realizando balanceos de carga de ser necesario.

## 2.1 Soluciones comerciales académicas

Sin duda alguna, con la incursión del las nuevas tecnologías y el cambio de paradigma respecto a las redes de comunicación tradicionales, SDN y SD-WAN abren el abanico a varias soluciones de ingeniería de tráfico. Es allí donde varios proveedores comenzaron a ver la importancia de las mismas y el que podrían llegar a alcanzar en caso de migrar su red hacia estas. La cantidad de diversas soluciones es muy amplia, pero a pesar de ello estas son en su totalidad servicios de pago, con lo cual no es posible realizar pruebas de laboratorio sin tener que pagar por ello. Algunas implementaciones cuentan con licencias trial pero están muy limitadas en cuanto a su uso y no permiten realizar lo que se espera en este trabajo.

A continuación se procederá a describir tres soluciones comerciales de servicio SD-WAN para entender mejor de que tratan, haciendo mayor hincapié en la solución de FlexiWAN ya que si bien es solución paga, esta es de código abierto, por lo que fue posible también realizar una serie de pruebas que se detallan en el punto 4.1 de este documento.

### 2.1.1 Nuege

**Nuage Networks** [i] es la solución SD-WAN implementada por Nokia que ofrece automatizar la conexión de sitios remotos en cualquier red con hardware. Permite entregar y organizar servicios de TI corporativos en centros de datos, nubes públicas o privadas y gestionar las conexiones WAN mediante la utilización de políticas optimizando así el uso de la red. Esto disminuye los costes de infraestructura satisfaciendo las necesidades requeridas al mismo tiempo que brinda seguridad a las aplicaciones.

Con VNS (*Virtualized Network Services*) de Nuage, la red SD-WAN se puede optimizar dinámicamente para enrutar tráfico por la red mas rentable. En muchos casos las empresas suelen tener redes IP, MPLS, 3G y/o LTE de diferentes proveedores para transportar sus servicios con lo cual se podría aprovechar por ejemplo internet para las aplicaciones que no requieran prestaciones exigentes dejando las demás conexiones para aquellos servicios críticos. El hecho de que estos enlaces sean provistos por diferentes

ISP es indiferentes para Nuage ya que es capaz de gestionarlos desde una plataforma unificada.

Por otra parte esta solución permite ocultar la complejidad de la red WAN empresarial estableciendo una WAN extremo a extremo que posibilita conectar centros de datos privados, sucursales y servicios de nube pública o privada como muestra la *Figura 2-1*.

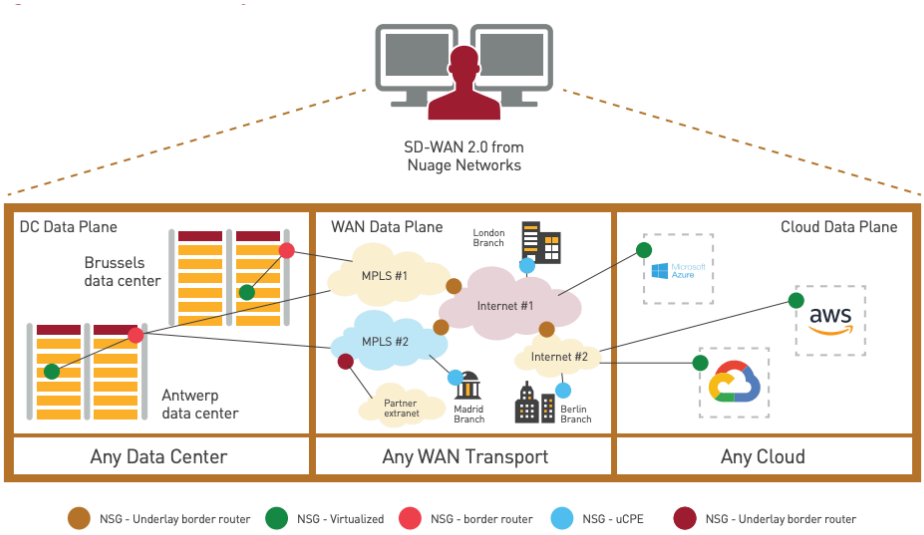



**Figura 2-1 – Red sin interrupciones SD-WAN**

La *Figura 2-2* muestra la arquitectura de Nuage. El VSD (Virtualized Services Directory) permite al administrados SD-WAN definir y aplicar políticas a la red de una manera fácil, gestionada y centralizada. A su vez implementa la funcionalidad de recolección de datos de forma periódica para la creación de informes y reportes de la red, así como también alertas en caso de que un tráfico haya sobrepasado un umbral determinado. Todas estas funciones se pueden desplegar en un portal personalizable con widgets.

Las funciones de red se pueden seleccionar mediante el catálogo de VSD, pudiendo optar entra VNF´s de firewalls, IPSec, NAT, balanceo de carga y gestión de dominios, entre otros.

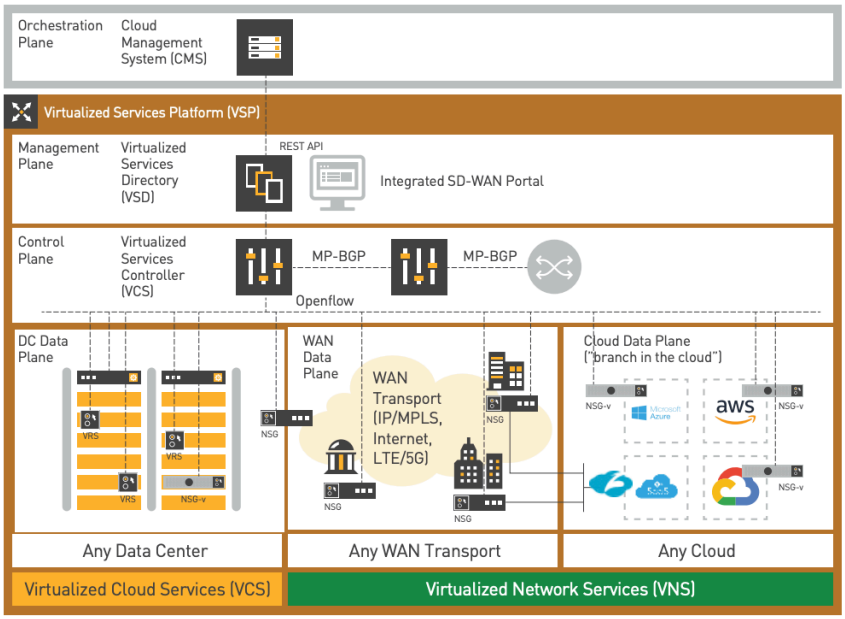



Figura 2-2 – Arquitectura Nuage

### 2.1.2 Viptela

**Viptela** [ii] es la solución SD-WAN adquirida por Cisco que permite abstraerse del medio de transporte y tener visibilidad en tiempo real del estado de la red para poder por ejemplo tomar decisiones de enrutamiento. Viptela está conformada por cuatro componentes, cada uno asociado a un plano diferente tal cual se ve en la *Figura 2-3*.

Por un lado se encuentra el *vEdge* que representa el router en las oficinas. Este es quien se encarga de establecer comunicaciones seguras con los demás vEdges (virtualizados o físicos) mediante túneles IPSec y son quienes ejecutan las políticas implementadas por el vSmart.

El *vSmart* es el controlador de la red encargado y es quien establece conexiones SSL con los demás componentes que forman la arquitectura SD-WAN mediante un protocolo propio llamado OMP (Overlay Management Protocol). Brinda seguridad, control de acceso y políticas de encaminamiento sin la utilización de protocolos tradicionales como OSPF y BGP. En cuanto a las políticas, estas pueden ser centralizadas aplicándose sobre todo el *Fabric* y actuando en varios sites, o bien, localizadas en un determinado vEdge.

El tercer elemento de la topología de Viptela es el dashboard centralizado bajo el nombre de *vManage*. Es el encargado de desplegar la información de la red el tiempo real y permitir al administrador aplicar reglas y monitorizar la red SD-WAN. Soporta diversos protocolos de gestión como SNMP, NETCONF o Syslog, así como también la capacidad que tienen los vEdges de realizar Deep Packet inspection le servirá al vManage para obtener analíticas a nivel de aplicación.

Por último, el *vBond* cumple el rol de orquestador SD-WAN y se encarga de realizar la autenticación y autorización de los elementos de la red gestionando las comunicaciones entre los vEdges y el controlador.

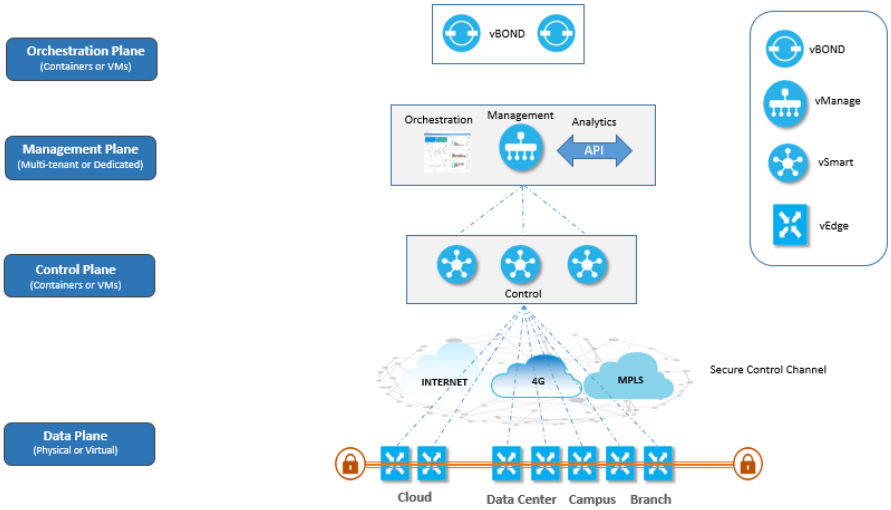



Figura 2-3 – Arquitectura Viptela

### 2.1.3 FlexiWAN

**FlexiWAN** [iii] es otra solución SD-WAN que a diferencia de las anteriores, es de código abierto la cual permite instalar hasta 3 instancias de forma gratuita aunque con ciertas limitaciones. La infraestructura SD-WAN de FlexiWAN está formada principalmente por dos actores principales tal como se ve en la *Figura 2-4*: *FlexiEdge* y *FlexiManage*. El FlexiEdge es el dispositivo que se ejecuta el punto de presencia el cual tiene incorporado VPP (Vector Packet Processor) y es utilizado para aumentar la velocidad de procesamiento ya que puede tratar varios paquetes el simultaneo. Por otro lado cuenta con FRR (Free Range Routing) que implementa diferentes tipos de protocolos de enrutamiento como OSPF, RIP, BGP, IS-IS, entre otros y se ejecuta en plataformas similares a Unix. Estos se conectan al FlexiManage, quien se encuentra ejecutándose sobre un servidor web y es quien administra de la red. A través de él un operario puede gestionar y recopilar información de los dispositivos FlexiEdge para luego analizarla y proporcionar informes de la red. Cabe destacar que la plataforma está fuertemente orientada a facilitar y centralizar las configuraciones y gestionar los equipos. Por tal motivo es que cuenta con funcionalidades capaces de desplegar alarmas y avisar al administrador de posibles fallos para que pueda actuar en el menor tiempo posible.

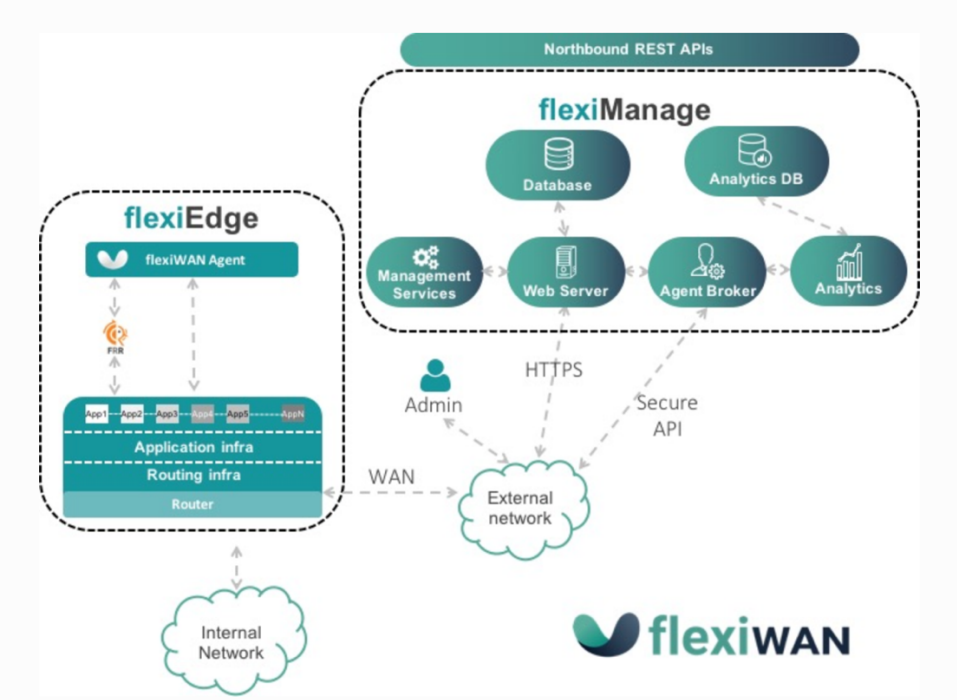



Figura 2-4 – Diagrama de bloques de alto nivel de la arquitectura FlexiWAN

## 2.2 Otros trabajos de investigación con ingeniería de tráfico SDN

Las soluciones planteadas en el apartado anterior muestran la diversidad de ideas a la hora de realizar soluciones de este tipo las cuales buscan tener un punto diferenciador respecto al resto. Sin embargo, en ellas no se deja del todo claro el aporte en cuanto a la ingeniería de tráfico (TE) que se aplica o si en definitiva lo hacen. A continuación se pasa a explicar algunos conceptos relevantes a la hora de realizar implementaciones TE, para luego introducir la solución SD-WAN de Google que busca mejorar la utilización de sus enlaces y comparte la idea general con la TE diseñada en este trabajo.

A rasgos generales la ingeniería de tráfico consiste en adecuar el tráfico a partir de las condiciones instantáneas de la red y la calidad de servicio que requieren las aplicaciones. Para lograrlo se busca cumplir con una serie de objetivos que se detallan a continuación.

- *Minimizar la congestión de red*: Mediante técnicas de denegación de acceso es posible impedir el acceso a los recursos congestionados, redistribuyendo los flujos mediante rutas multicaminos.

- *Reducir el retardo extremo a extremo*: Técnicas como CSFP (*Constrained Shortest Path First*) permite determinar el camino lógico mas corto entre dos puntos de la red, minimizando así el retardo.

- *Minimizar la pérdida de paquetes:* Se puede dar por diversos motivos como caída o saturación de los enlaces. Para disminuir este factor se suelen emplear caminos de respaldo o protocolos de convergencia para disminuir al máximo el problema.

- *Priorización de tráfico:* Es posible realizar técnicas para determinar diferentes niveles de tráfico y así tratarlos de forma diferenciada. Esto hace que se envíe un tipo de paquete mas sensible antes que otro, lo cual puede traer como consecuencia una mejoría en la calidad de experiencia de dicho servicio. El caso mas sencillo de visualizar es para el tráfico de streaming donde se requiere que este sea lo mas fluido posible.

Si bien mediante el encaminamiento en redes MPLS es posible cumplir con algunas de estos puntos, sigue habiendo algunas limitantes para cumplirlos en su totalidad y es aquí donde implementaciones basadas en SDN vienen a solucionar estos temas. Ejemplo de esto es el inconveniente que trae asociado MPLS-TE a la hora del congestionamiento en escenarios multicamino. Aquí se requiere de una reordenación excesiva de paquetes y posible desorden en el destino que puede llegar a acabar en un problema de congestión avoidance, produciendo degradación de throughput de las aplicaciones.

Existen diversas soluciones de ingeniería de tráfico SDN orientadas a alcanzar los objetivos anteriormente mencionados. Ejemplos de estas son B4 o SWAN propuestas por Google y Microsoft respectivamente. Estas comparten algunas características pero difiere en otras. SWAN define clases de tráfico y a cada una de ellas le asigna un ancho de banda determinado en función de las prestaciones que requieren. Por otra parte, B4 se basa en el principio de reparto justo, el cual consiste en dividir los recursos de red disponibles entre las diferentes aplicaciones en orden creciente de demanda. Esta última solución es la que mas se asemeja a lo que se busca en dicho trabajo ya que trabaja sobre una red totalmente SDN, con lo cual se pasa a detallar mas en profundidad en el siguiente apartado 2.2.1.

### 2.2.1 B4

**B4** [iv] es la solución de SD-WAN implementada por Google que se basa en la interconexión dos sitios remotos mediante una red SDN, es decir, utilizando conmutadores OpenFlow. La idea primaria de B4 es realizar una ingeniería de tráfico capaz de implementar algoritmos de encaminamiento en una red multicamino teniendo en cuenta los requisitos de QoS de las aplicaciones. En segundo lugar, adaptar la asignación de ancho de banda y las demandas variables de las aplicaciones y posibles caídas en los enlaces y conmutadores.

El algoritmo permite determinar los túneles virtuales que se deben programar en la red para que se cumplan con los requisitos de las aplicaciones, así como también el ancho de banda asignado a cada una. Este es asignado en función de las prioridades de

las aplicaciones las cuales vienen dadas a partir del volumen y la sensibilidad frente a retardos.

# 3 Casos de uso

## 3.1 Caso de uso A

Este trabajo se basa principalmente en la idea de interconectar dos sedes remotas de un empresa. A modo práctico se supone que se desea conectar una oficina central ubicada en la ciudad de Madrid con la sucursal mas pequeña que se encuentra en Barcelona. Esta última no cuenta con centro de datos propio sino que todas las aplicaciones y repositorios se encuentran alojados en el sitio central, por lo cual se necesita tener una óptima conexión entre sitios de forma que los trabajadores en Barcelona tengan la misma calidad de experiencia y servicio como si se estuviera en Madrid. A raíz de ello la empresa en cuestión tiene contratados 2 enlaces WAN. Uno, a cuyo proveedor lo llamaremos *proveedor A* que brinda conectividad a internet, mientras que el *proveedor B* proporciona un enlace privado en su red SD-WAN el cual permite interconectar ambos sitios proporcionando una determinada calidad de servicio. Este último su vez debe ser capaz de brindar unos estándares de forma de cumplir con el contrato firmado. Para ello debe aplicar soluciones de ingeniería de tráfico (TE) a nivel de su red de forma de encaminar los paquetes más prioritarios de una manera óptima en función del estado de la misma. Aquí es donde aparece reflejado el primer caso de uso ya que una solución de este tipo aumenta la calidad de la red, la experiencia del usuario y disminuye los costos de mantenimiento y despliegue ya que se hace un mejor uso de la red.

Por otra parte, una solución así permite una granularidad tal que posibilita un servicio muy orientado a las necesidades del cliente. Si bien se entiende de por sí que los paquetes que se cursan por esta red requieren de cierta calidad de servicio, a su vez, dentro de la red SD-WAN permite darle aún mas prioridad. Si bien esta solución de TE es transparente para el usuario final, este se ve directamente beneficiado ya que puede cursar los paquetes de una forma ágil, cumpliendo con los requisitos de las aplicaciones y a un bajo costo.

La red del proveedor B suele ser compleja ya que busca interconectar varios sitios remotos de diversas empresas. Para dificultar el caso, se asume que tiene una topología de pez, la cual se explica con mayor detalle en la sección 4.3 de este documento. Este es el caso que se trata en este trabajo.

Por otra parte, la empresa puede tener contratado a su vez los servicios del *proveedor C* el cual le ofrece una solución de ingeniería de tráfico en tiempo real a la empresa

(cliente final) que permite monitorear en estado de los enlaces WAN y realizar reglas para un mejor tratamiento del tráfico entre oficinas decidiendo bajo demanda cual de las redes WAN se debe utilizar. Una buena práctica sería enviar todo el tráfico que no requiere de altas prestaciones por el enlace de internet, cursando aquel mas sensible por la red SD-WAN del proveedor B.

Como último punto a mencionar es que los tres proveedores pueden ser distintos aunque esto no tiene por qué ser obligatoriamente así, siendo el mismo proveedor quien brinde mas de un servicio a al empresa. SDN permite una diversidad de soluciones, proporcionadas por una diversidad de proveedores, algo que hasta el momento con las redes tradicionales no era posible.

Este escenario recientemente explicado se muestra en la *Figura 3-1.*

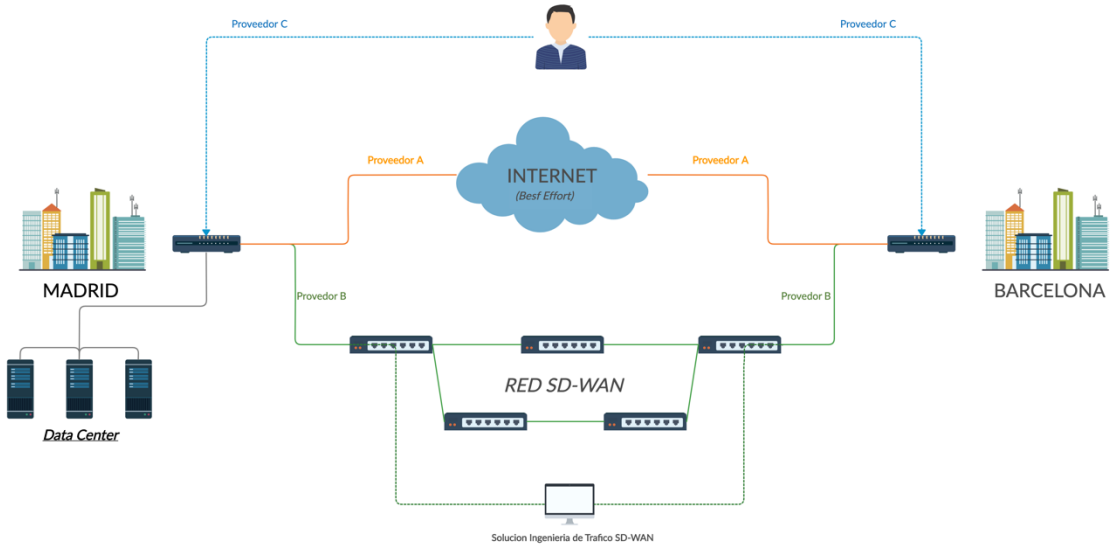



**Figura 3-1 – Escenario modelo de caso de uso 1**

## 3.2 Caso de uso B

El segundo caso de uso donde se puede aplicar este trabajo es a la hora de interconectar dos centros de datos mediante una red SD-WAN multicamino. En este caso se supone que los caminos tienen diferente métrica con lo cual de no se aplicarse soluciones TE, se haría una uso indebido de los enlaces dejando algunos sin utilizar. Esta solución no implica necesariamente que el proveedor de la red SD-WAN sea el mismo que el de los centros de datos.

Se supone un escenario como se detalla en la *Figura 3-2* donde hay dos CPD´s interconectados, consumiendo uno recursos del otro. A su vez el flujo de datos entre ellos puede ser muy variado, haciendo que estos no sean igualmente prioritarios sino que hay servicios que requieren de mayores prestaciones que otros. Para este caso es necesario aplicar una solución de ingeniería de tráfico de forma de priorizar algunos

tipos de datos, ya sea teniendo en cuenta por ejemplo direcciones MAC, IP o puertos de origen y/o destino. Mediante esta asignación de prioridades se debe orientar el tráfico por el camino correcto de forma de asegurar el nivel de servicio requerido para el funcionamiento de las aplicaciones. Es aquí donde aparece la necesidad de aplicar TE.

De forma centralizada el operario debe poder seleccionar el o los tipos de tráficos sensibles, los umbrales de anchos de banda necesarios para el funcionamiento y las políticas a aplicar en los conmutadores para cumplir con estas exigencias. A su vez, esto será gestionado de forma automática por el controlador luego se que el operador haya realizado las configuraciones ya que de no ser así se haría poco escalable.

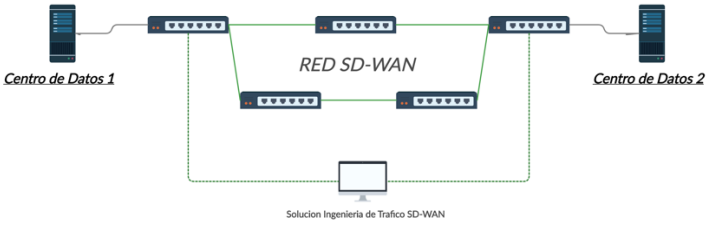



Figura 3-2 – Escenario modelo de caso de uso 2

# 4 Aplicación de ingeniería de tráfico en SD-WAN

En esta sección se procede a detallar en profundidad la solución implementada, las decisiones tomadas, las herramientas utilizadas y las pruebas técnicas realizadas para dejar constancia del alcance de la misma.

En problema a abordar consiste en el desarrollo e implementación de una solución de ingeniería de tráfico SD-WAN que sea capaz de poder seleccionar bajo demanda el camino que debe tomar tráfico determinado a partir de dos componentes esenciales: estado de la red en tiempo real y parámetros de QoS. El escenario a tratar (sección 4.3) busca parecerse lo mas posible a la realidad (sección 3.1) donde para llegar de un origen a un destino puede haber mas de un camino posible y es aquí donde el controlador debe decidir por cual de ellos debe enviar los paquetes. Se entiende por "tráfico sensible" todo aquel que previamente haya definido el administrador de red como tal y este se especifica en el apartado 4.5.

## 4.1 Pruebas preliminares sobre servicios SD-WAN de código abierto

Una vez estudiada la teoría de FlexiWAN [iii] en el punto 2.1.3 de este documento, se procedió a implementar un escenario para ver en funcionamiento de las prestaciones que ofrece la plataforma en la versión gratuita. En primer lugar se crearon y configuraron en VirtualBox dos máquinas virtuales Ubuntu con conexión a internet para que tengan conectividad con el FlexiManage. Estas dos instancias simulan ser dos sitios remotos.

En ambas fue necesario asociarles un token de autenticación (creado en el FlexiManage) para la autenticación y establecer una canal de comunicación contra ellos. Una vez hecha esto, los routers virtuales aparecen en el dashboard dentro de la sección Inventory/Devices and manage tal cual lo indica la *Figura 4-1* como "Running".

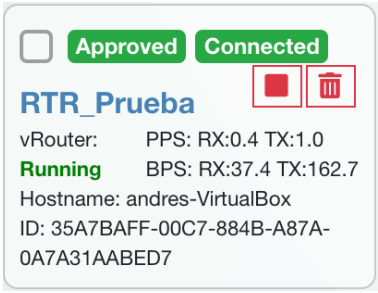



**Figura 4-1 – Visualización el FlexiWAN-router desde la interfaz central manage**

Una vez eniendo los sitios remotos configurados y activos en el FlexiManage se puede comenzar a realizar las visualizaciones y configuraciones que se deseen como por ejemplo cambiar de forma remota las direcciones IP públicas y privadas del router, así como también crear nuevas interfaces. Todo esto se hace de forma centralizada y en pocos minutos lo cual proporciona una flexibilidad y rapidez en la solución difícil de igualar en una red WAN tradicional. Por otro lado se crearon túneles virtuales entre los FlexiEdges de forma de darle conectividad entre ellos. Una vez hecho esto la herramienta permite medir la latencia y los paquetes perdidos en dicho canal. Esto se hace también de forma centralizada desde la sección Inventory/Tunneles, lo cual muestra otra vez una gran ventaja respecto a sistemas tradicionales donde es necesario realizar configuraciones en ambos extremos para establecer un túnel.

En cuanto a las visualizaciones que permite la plataforma, estas son bastante limitadas pero aceptables. Por un lado es posible obtener las tablas de ruteo de cada FlexiEdge y modificarlas o agregar nuevas rutas de ser necesario. También posibilita la obtención de estadísticas en tiempo real del tráfico cursado tanto en BPS (bits por segundo) o PPS (paquetes por segundo) como muestra en la *Figura 4-2*.

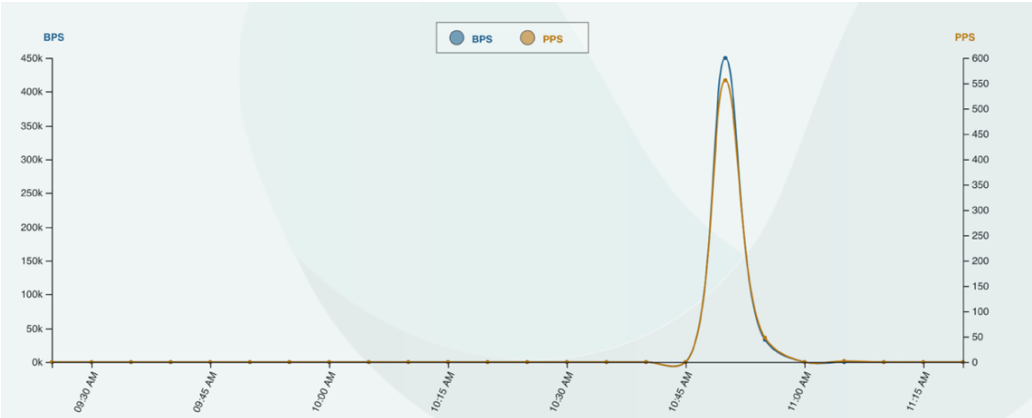



**Figura 4-2 – Estadísticas de bps y pps en un FlexiWAN-router**

A pesar de todas estas pruebas que se realizaron, se ve que si bien busca centralizar la gestión de red en un dashboard interactivo, sigue siendo es una herramienta bastante hermética y no posibilita la generación de políticas de calidad ni mucho menos de ingeniería de tráfico.

## 4.2 Controlador Ryu

SDN cuenta con una arquitectura muy diferente a la tradicional ya que se busca desacoplar de forma lógica y en algunos casos física el plano de control [1] del plano de datos [2] haciendo así la red mas programable, automatizada y controlable. A raíz de esto nace el concepto de controlador, el cual oficia como "cerebro" de la red teniendo una visión general de la misma y comunicándose con los equipos mediante protocolos estandarizados como por ejemplo OpenFlow. Este al ser estándar permite gestión y control centralizado de equipamiento multivendor, utilización de API´s comunes y control granular mediante políticas de sesión, usuarios, dispositivos y aplicaciones.

En la *Figura 4-3* [v] se puede apreciar una visión lógica de este tipo de arquitecturas SDN donde queda en evidencia el desacople del plano de datos (capa de infraestructura) y el plano de control (capa de control). En este último se encuentra el controlador SDN el cual se comunica en "sentido norte" a la capa de aplicaciones mediante API´s haciendo así muy gestionable y programable en función del uso requerido. También permite la integración con aplicaciones de negocio, calidad de servicio, entre otras. Por otro lado, la comunicación hacia los dispositivos de red, comúnmente denominados conmutadores, se hace como se mencionó anteriormente por medio de protocolos estándares como OpenFlow mediante paquetes *packet-in* [3] y *packet-out* [4] principalmente. Mediante estos, el controlador se comunica con los routers y switches SDN permitiendo manipular por ejemplo sus tablas de forwarding ya que estos delegan su inteligencia al controlador y pasan a ser simples unidades de conmutación de red.

[1] Orientado a tratar el tráfico orientado a la gestión, mantenimiento y modificación de la red y sus componentes.
[2] Orientado a tratar el tráfico destinado a los servicios.
[3] Paquetes del conmutador al controlador.
[4] Paquetes del controlador al conmutador.

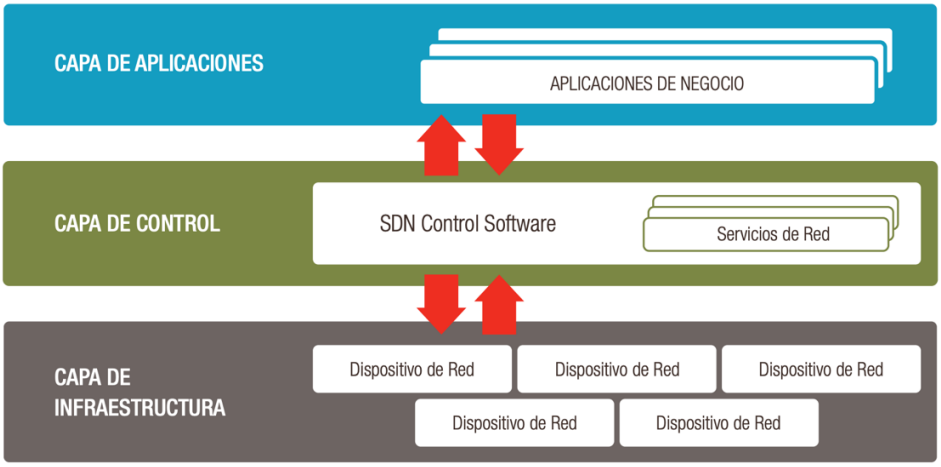



**Figura 4-3 – Arquitectura SDN**

El controlador elegido para esta implementación es Ryu [5] debido a una serie de factores que lo diferencian del resto. En primer lugar es una solución Open Source SDN diseñada para incrementar la agilidad de la red facilitando la administración y adaptación de cómo se maneja el tráfico [vi]. Cuenta con una gran cantidad de aplicaciones creadas por desarrolladores y publicadas en diversos portales abiertos y foros reconocidos. De esta forma se potencializa el uso de la plataforma ya que se genera un clima acorde para la transferencia de conocimientos y potenciamiento de la herramienta, aumentando así el alcance de la misma. En segundo lugar soporta NETCONF, extensiones Nicira y diversas versiones de OF (OpenFlow) [vi], en especial la 1.3 que es la utilizada en este trabajo. Provee una API REST básica pero a la vez poderosa que ayuda a diseñar soluciones acordes para cada desarrollador, a parte de una interfaz web muy limitada pero que al ser de código abierto permite su adaptación.

Este framework está escrito en lenguaje Python y apoyado sobre plataforma Linux lo cual permite la virtualización con Mininet y OpenvSwitch. Esta característica permite a los administradores crear redes virtuales de forma dinámica, disociadas de las redes físicas para satisfacer la ampliación de la capacidad sin afectar los flujos existentes [vii].

Otro motivo por el cual se opta por este controlador es que permite configurar flujos de forma proactiva y reactiva. La primera consiste en configurar la regla de flujo antes de que el paquete llegue al switch OpenFlow. De esta forma una vez que un paquete ingrese al mismo, éste ya conoce el tratamiento que debe realizar, reduciendo así el tiempo de encaminamiento. Esto hace mediante la API REST propia de Ryu. Por otro lado, una configuración reactiva se produce cuando el conmutador recibe un paquete que no coincide con ninguna de sus entradas en su tabla de flujo y debe realizarle una consulta mediante packet-in al controlador, respondiéndole éste con un

[5] https://ryu-sdn.org/index.html

packet-out la regla de como tratarlo. La desventaja del procedimiento reactivo frente al proactivo es que a gran escala éste genera un tráfico considerable en la red así como también un retardo (tiempo de paquetes del controlador al conmutador y viceversa, tiempos de procesamiento de los paquetes y tiempo de llenado de tablas de flujo) [vii].

El código del controlador Ryu se encuentra configurado el Python y está alojado en Github [6]. La comunidad Ryu es quien se encarga de efectuarle mejoras, solucionando los posibles fallos y agregando nuevas funcionalidades.

### 4.2.1 Traffic Monitor

Conjuntamente con el controlador Ryu se ejecuta el script de Traffic Monitor [7] el cual permite obtener en tiempo real el estado de los enlaces de la red de una forma rápida y sencilla. Cabe destacar que la versión utilizada en este trabajo es una extensión modificada del controlador de código abierto creada por la comunidad de programadores, la cual permite visualizar a nivel de consola los links de los conmutadores, el estado de los mismos, el tráfico instantáneo cursado por las interfaces y un histórico de la cantidad de paquetes y bytes enviados, recibidos y con errores.

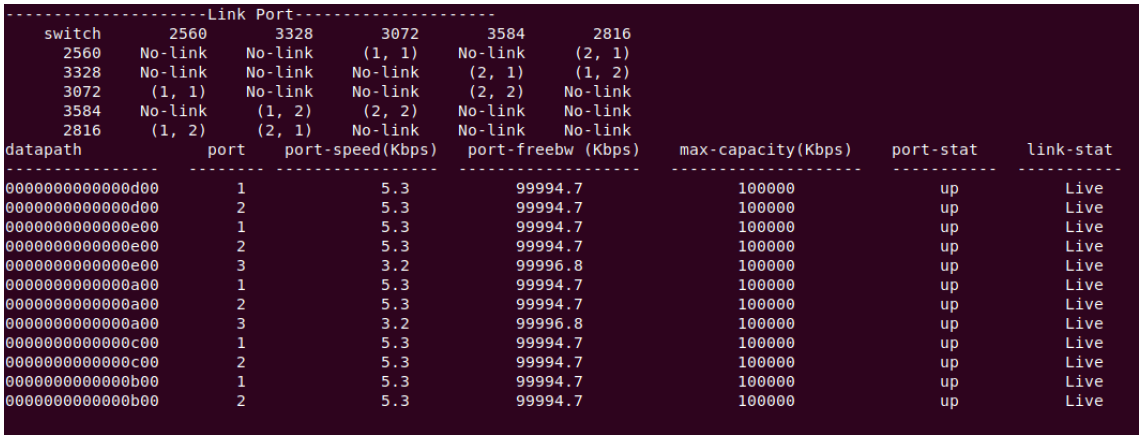
Por otro lado, existe una instancia que al ejecutarse por primera vez, realiza un escaneo rápido de la red y obtiene la topología de red, viendo así las conexiones entre conmutadores e imprimiéndolas en pantalla tal cual se ve en la sección "Link Port" de la *Figura 4-4*. A modo de ejemplo se puede ver que el conmutador con datapath *2560* (CONM_A) se encuentra conectado por el puerto 1 al puerto 1 del *3072* (CONM_C) y por el puerto 2 al puerto 1 del *2816* (CONM_B). De forma análoga se ven el resto de las conexiones. Esta nomenclatura tan particular la obtiene a partir de convertir en decimal la dirección *hwaddr* de los conmutadores. A modo de ejemplo para el conmutador A (CONM_A) el valor del hwaddr es 0x000000000A00, el cual equivale a 2560 en base decimal.

```
1        <net name="CONM_A" mode="openvswitch" hwaddr="00:00:00:00:0A:00" controller="tcp:127.0.0.1:6633"
of_version="OpenFlow13" fail_mode="secure" >
```

**Código 4-1 – Extracto de configuración de CONM_A**

La instancia del controlador se encuentra escrita puramente en lenguaje Python y en este caso fue modificado para adaptarlo a lo que se requería en el trabajo. En primer lugar se quitaron las columnas de paquetes enviados, recibidos y con errores ya que a efectos de las prácticas a realizar, estos valores no interesan. Por otra parte, se agregó una columna nueva que indica la máxima capacidad del enlace llamada *max-capacity* expresada en Kbps la cual indica el ancho de banda del enlace. A efectos de simplificar el escenario se configuraron todos en 100 Mbps.

[6] https://github.com/faucetsdn/ryu
[7] https://github.com/muzixing/ryu



```
--------------------Link Port--------------------
    switch       2560       3328       3072       3584       2816
      2560    No-link    No-link     (1, 1)    No-link     (2, 1)
      3328    No-link    No-link    No-link     (2, 1)     (1, 2)
      3072     (1, 1)    No-link    No-link     (2, 2)    No-link
      3584    No-link     (1, 2)     (2, 2)    No-link    No-link
      2816     (1, 2)     (2, 1)    No-link    No-link    No-link
datapath             port    port-speed(Kbps)   port-freebw (Kbps)    max-capacity(Kbps)    port-stat     link-stat
---------------    -------- ----------------  ------------------    ------------------    -----------  -----------
0000000000000d00        1              5.3           99994.7              100000              up           Live
0000000000000d00        2              5.3           99994.7              100000              up           Live
0000000000000e00        1              5.3           99994.7              100000              up           Live
0000000000000e00        2              5.3           99994.7              100000              up           Live
0000000000000e00        3              3.2           99996.8              100000              up           Live
0000000000000a00        1              5.3           99994.7              100000              up           Live
0000000000000a00        2              5.3           99994.7              100000              up           Live
0000000000000a00        3              3.2           99996.8              100000              up           Live
0000000000000c00        1              5.3           99994.7              100000              up           Live
0000000000000c00        2              5.3           99994.7              100000              up           Live
0000000000000b00        1              5.3           99994.7              100000              up           Live
0000000000000b00        2              5.3           99994.7              100000              up           Live
```

Figura 4-4 – Tablas de topología y puertos por tráfico del programa Traffic Monitor de Ryu

Las columnas más relevantes de la *Figura 4-4* son las llamadas *port-speed (Kbps)* y *port-freebw (Kbps)* las cuales devuelven en tiempo real el tráfico cursado por cada interfaz y el ancho de banda disponible en cada una respectivamente. Estos campos se rigen mediante la siguiente ecuación:

$$portFreebw\ (Kbps) = maxCapacity\ (Kbps) - portSpeed(Kbps)$$

Ecuación 4-1 – Ecuación para la obtención del ancho de banda disponible

Por último se agregó una línea de código en el Traffic Monitor la cual cada 10 segundos exporta los valores de la tabla de la *Figura 4-4* a un fichero csv llamado "*Table.csv*". Esto permitirá hacer un tratamiento con estos datos según se detalla en la sección 4.4 de este documento.

### 4.2.2 Ofctl_rest

Mediante la interfaz northbound el conmutador Ryu es capaz de interactuar con la API ofctl_rest [viii]. Esto es esencial para el desarrollo de la solución en cuestión ya que mediante métodos POST es posible agregar, modificar o quitar entradas en las tablas de flujo de los conmutadores. La secuencia de la misma se detalla a continuación y se puede acompañar con la *Figura 4-5*:

1) El usuario solicita por ejemplo agregar una entrada de flujo en un conmutador. Para ello establece una conexión TCP hacia el puerto 8080 donde se encuentra escuchando la API. Estos intercambian una serie de paquetes de sincronización TCP para luego realiza el POST HTTP. Cerrando por último la conexión.

2) El controlador realiza la función de pasarela, enviando mediante un paquete OpenFlow de tipo FLOW_MOD [8] la acción hacia el puerto TCP 6633 que tiene escuchando hacia el controlador.

[8] Tipo de paquete OpenFlow que permite modificar el estado de un switch OpenFlow

3) El conmutador modifica su tabla de flujo con la entrada configurada en el punto 1.

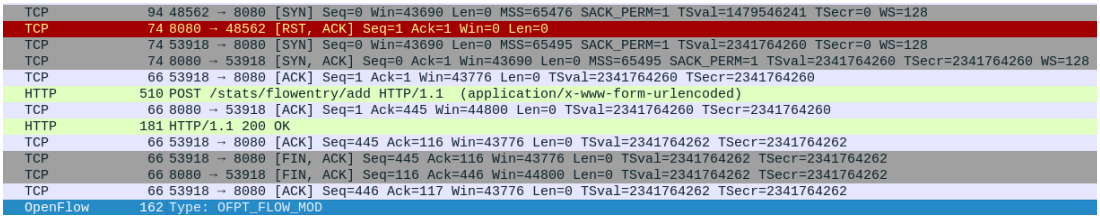



**Figura 4-5 – Captura de tráfico ante solicitud de modificación de entrada de flujo hecha con Wireshark**

Los POST a la API de Ryu se hacen mediante json como se indica en el siguiente ejemplo perteneciente a un fragmento de código del script *Sensible_traffic_long_path.sh.*

```
1   curl -X POST -d '{
2           "dpid": 2560,
3           "cookie": 0,
4           "cookie_mask": 1,
5           "table_id": 0,
6           "priority": 5,
7           "flags": 1,
8           "match":{
9                   "in_port": 3,
10                  "tcp_dst": 22,
11                  "ip_proto": 6,
12                  "eth_type": 2048
13          },
14          "actions":[
15                  {
16                          "type":"OUTPUT",
17                          "port": 2
18                  }
19          ]
20  }' http://localhost:8080/stats/flowentry/add
```

**Código 4-2 – Extracto de código de script "Sensible_traffic_long_path.sh**

Las línea 2 del *Código 4-2* indica el identificador del conmutador donde se desea aplicar la regla de tráfico. Seguido de esto, entre la 3 y la 7 de especifican una serie de parámetros OpenFlow donde para este trabajo el mas relevante de ellos es el campo "priority". El resto se encuentra con los valores por defecto. De la línea 8 a la 13 indican que si el paquete entrante coincide con todos esos parámetros (puerto de entrada en el conmutador, número de puerto TCP destino, tipo de protocolo IP y ethernet), entonces se lo trata de acuerdo a lo referida entre las líneas 14 y 19. Para este caso la acción consiste en despachar le paquete por el puerto 2 del conmutador. Por último, la línea 20 de este código corresponde la URL a la cual se hace el POST.

## 4.3 Escenario de red virtual

El escenario planteado para la realización de este trabajo fue elaborado mediante la herramienta de virtualización de redes VNX el cual permite levantar un fichero xml con la configuración del mismo. A la hora de su diseño se tuvo en cuenta principalmente dos condiciones:

- Más de un camino posible para llegar de un origen a un destino.
- Red multicamino con **topología de pez**.

Estas dos premisas son importantes para poder luego poder implementar la ingeniería de tráfico del objetivo, la cual permita decidir en tiempo real el camino que deben tomar los paquetes. En cuanto a la topología de pez esta se puede apreciar en la *Figura 4-6* con los conmutadores A,B,C,D y E. Aquí se puede ver que para encaminar un paquete desde R1_edge a R2_edge por la red de conmutadores es posible hacerlo por dos caminos diferentes. Uno pasando por CONM_A, CONM_C y CONM_E, al cual llamaremos de aquí en más "short path" y otro conformado por CONM_A, CONM_B, CONM_D y CONM_E al cual se lo llamará "long path". Lo interesante aquí no radica en la existencia de multicaminos sino en que uno de ellos es mas largo que el otro. Este problema es bien conocido y deja al descubierto un punto débil de los protocolos de ruteo tradicional como RIP u OSFPF ya que trata automáticamente un escenario multicamino como si fuera de uno solo.

Por otro lado existe también una red llamada "INTERNET" que oficia de enlace best effort por el cual iría el tráfico que no se requiere calidad de servicio. Para este trabajo si bien se implementó en el escenario para hacer que sea lo mas parecido a la realidad, no se tiene en cuenta y no hay tráfico por el ya que los routers están configurados para que encaminen los paquetes hacia la red de conmutadores que es con la que interesa trabajar en este caso.

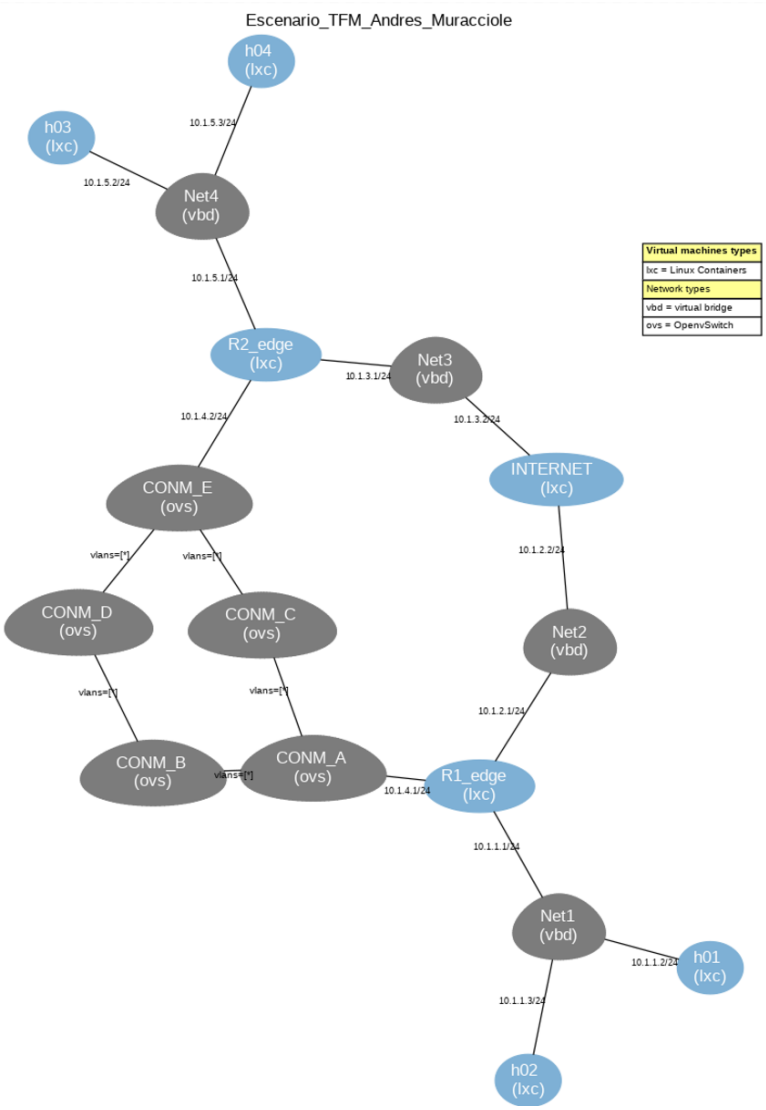



Figura 4-6 – Escenario virtual con red de conmutadores en topología de pez

Volviendo a la red de conmutadores, en un escenario con ruteo clásico y suponiendo como es el caso, que los enlaces son todos iguales, el encaminamiento se dará por la ruta mas corta, como se puede ver de forma ilustrativa en la *Figura 4-7*. Esto se debe a que tiene menor métrica, inutilizando así el camino mas largo. De esta forma se puede llegar a saturar la red rápidamente a pesar de tener un camino por el cual no se está traficando datos [ix].

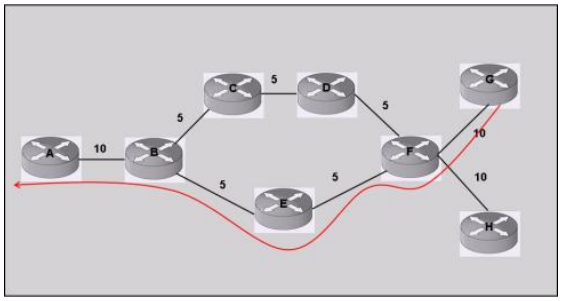



Figura 4-7 – Topología de pez

Hasta el momento la forma que se tenía de poder utilizar el camino largo es asignando el costo de los enlaces de forma manual, haciendo así que el costo total final del camino largo sea menor que del corto como se muestra en la *Figura 4-8*. Sin embargo esto ni aporta a resolver el problema ya que lo que genera es una migración del tráfico de un camino al otro dejando ahora el camino corto sin utilización y volviendo a tener congestión en la red.

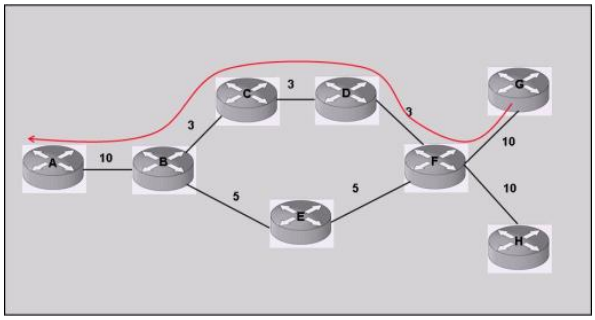



**Figura 4-8 – Topología de pez**

Si bien una forma de solucionar esto es asociando manualmente los costes de los enlaces de tal forma que para ambos caminos sea igual y de esta forma utilizar los dos. En esta trabajo no se busca resolver el balanceo de carga de dicha forma sino que se hace dinámicamente mediante entradas en las tablas de flujo de los conmutadores. De esta forma se puede no solo hacer un uso mas eficiente de la topología de red sino que también permite al administrador especificar el tipo de tráfico que irá por cada enlace.

La *Figura 4-6* muestra el escenario realizado en VNX el cual está conformado por los siguientes componentes y configurados como se muestra en la *Tabla 4-1*:

- 4 contenedores LXC que ofician de sistemas finales: h01, h02, h03 y h04
- 5 conmutadores OVS: CONM_A, CONM_B, CONM_C, CONM_D y CONM_E
- 2 contenedores LXC-Vyos que ofician de routers de sedes remotas: R1_edge y R2_edge
- 4 redes: Net1, Net2, Net3 y Net4
- 1 contenedor LXC que representa el enlace a internet: INTERNET

| NOMBRE | INTERFAZ | MAC | IPv4 |
|---|---|---|---|
| h01 | 1 | 00:00:00:00:01:01 | 10.1.1.2 |
| h02 | 1 | 00:00:00:00:02:01 | 10.1.1.3 |
| h03 | 1 | 00:00:00:00:03:01 | 10.1.5.2 |
| h04 | 1 | 00:00:00:00:04:01 | 10.1.5.3 |
| R1_edge | 1 | 00:00:00:00:06:01 | 10.1.1.1 |
| R1_edge | 2 | 00:00:00:00:06:02 | 10.1.2.1 |
| R1_edge | 3 | 00:00:00:00:06:03 | 10.1.4.1 |
| R2_edge | 1 | 00:00:00:00:07:01 | 10.1.5.1 |

| | | | |
|---|---|---|---|
| R2_edge | 2 | 00:00:00:00:07:02 | 10.1.3.1 |
| R2_edge | 3 | 00:00:00:00:07:03 | 10.1.4.2 |
| INTERNET | 1 | 00:00:00:00:05:01 | 10.1.2.2 |
| INTERNET | 2 | 00:00:00:00:05:02 | 10.1.3.2 |
| CONM_A | - | 00:00:00:00:0A:00 *(hwaddr)* | - |
| CONM_B | - | 00:00:00:00:0B:00 *(hwaddr)* | - |
| CONM_C | - | 00:00:00:00:0C:00 *(hwaddr)* | - |
| CONM_D | - | 00:00:00:00:0D:00 *(hwaddr)* | - |
| CONM_E | - | 00:00:00:00:0E:00 *(hwaddr)* | - |

**Tabla 4-1 – Direccionamiento L2 y L3 del escenario**

Los conmutadores se conectan al controlador Ryu mencionado en la sección 4.2 del documento mediante el protocolo OpenFlow 1.3.

### 4.3.1 Wondershaper

Una forma sencilla de poder limitar en Linux el ancho de banda de las interfaces es mediante el programa Wondershaper [9] el cual permite asignar de forma individual un límite de subida y bajada expresado en Kbps. Para este trabajo se creó un script bash el cual se ejecuta de forma automática al levantar el escenario y determina un ancho de banda máximo de 100 Mbps en las interfaces "A-C-0", "A-B-1", "D-E-1" y "E-C-0", de forma que se asegura que no habrá mas de este ancho de banda en la red SDN.

## 4.4 Desarrollo de aplicaciones SDN

A continuación se procede a explicar el contenido central de la solución de ingeniería de tráfico implementada de forma detallada. Ésta se encuentra totalmente automatizada y centralizada ya que el usuario solo debe abrir la aplicación ejecutable "TE_SD-WAN" el cual despliega un menú mediante el cual el usuario interactúa con el. En backend se van instanciando los demás códigos (escritos en Bash o Python según corresponda) de forma transparente ante la vista del operario.

Todo lo relacionado a esta solución de ingeniería de tráfico se encuentra disponible de forma pública en el repositorio Github del autor [10].

Como se mencionó recientemente, la solución comienza iniciando al script central el cual despliega en pantalla el menú interactivo de la *Figura 4-9.* Aquí el usuario tiene la posibilidad de realizar varias acciones que se detallan en los siguientes apartados.

[9] https://www.tecmint.com/wondershaper-limit-network-bandwidth-in-linux/
[10] https://github.com/amuracciole/TrafficEngineering_SDWAN.git


Figura 4-9 – Menú principal interactivo

### 4.4.1 OPCION 1: Crear escenario

Este paso es el primero que se debe realizar. Para aplicar una solución de tráfico, primero es necesario desplegar el escenario al cual se le aplicará. En este caso es un escenario virtualizado en VNX introducido en el punto 4.3, con lo cual al poner la opción 1, este ejecuta el escenario (caja A) y pone en funcionamiento dos scripts (cajas B y C) los cuales mediante la herramienta Wondershaper se definen los anchos de banda de los canales de la red de conmutadores en 100 Mbps de uplink y downlink. La secuencia de estos pasos se ven reflejados en el diagrama de la *Figura 4-10*.

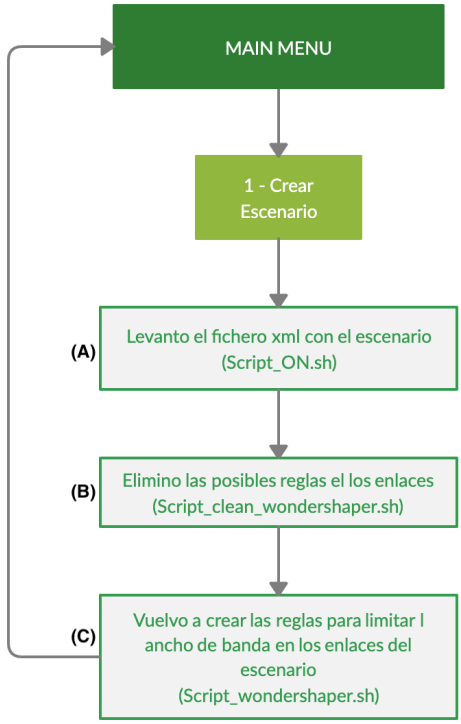



Figura 4-10 – Diagrama. OPCION 1) Crear Escenario

Una vez que se termina de crear el escenario correctamente, se devuelve al menú principal para que el usuario continúe interactuando con el.

### 4.4.2 OPCIÓN 2: Destruir escenario

La opción número 2 realiza la secuencia de la *Figura 4-11*, la cual destruye el escenario creado en el apartado 4.4.1.

Llama al script *OFF.sh* (caja D) que borra la configuración VNX. Esta opción es importante ya que al crearse un escenario virtual, en caso de no destruirlo manualmente, éste permanece guardado en la memoria del computador, lo cual puede traer problemas si se intenta volver a levantar uno ya que detecta la existencia de componentes con el mismo nombre y el escenario queda corrupto.

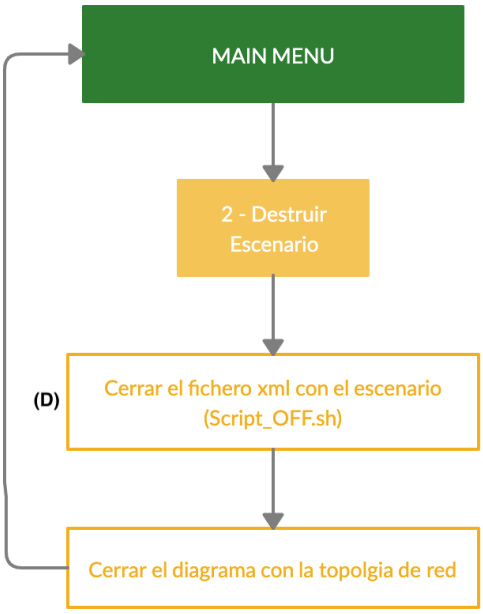



Figura 4-11 – Diagrama. OPCION 2) Destruir Escenario

### 4.4.3 OPCION 3: Ejecutar controlador

La opción de "Ejecutar controlador" es el eje de la solución ya que es la encargada de encender el controlador Ryu con el Traffic Monitor modificado (caja E), definir los caminos iniciales del tráfico ARP e IP (cajas F y G respectivamente) y llamar al script *Monitor.py* (caja H) como se ve en el diagrama de flujo de la *Figura 4-12*. Este último se encuentra constantemente censando los enlaces de los conmutadores.

En la caja E se inicializa el controlador Ryu con el Traffic Monitor, el cual permite por un lado conectar a él los conmutadores y por otro, imprimir cada 10 segundos el estado de la red en pantalla. Esta información será de vital importancia para luego poder decidir el camino que tomaran los paquetes.

Al iniciar por primera ver el escenario y por como está diseñada la solución, el controlador no tiene la potestad de elegir el mejor camino por le cual enviar los paquetes en una red de anillo. Por ello es que es necesario forzar manualmente un camino por defecto por el cual enviar el tráfico entre los routers edge de la red. Aquí entran en juego las cajas F y G los cuales son shell scripts formados por una serie de POST de json a la API que permiten configurar el controlador por la interfaz northbound. Estos hacen que Ryu escriba las entradas de flujo necesarias en los conmutadores CONM_A, CONM_C y CONM_E y así permitir enrutar todo el tráfico IPv4 y ARP por ellos. De esta forma se evitan los loops ya que se define un único camino para dicho tráfico. Esta configuración es por defecto pero luego se verá que se puede cambiar tanto de forma manual como automatizada.

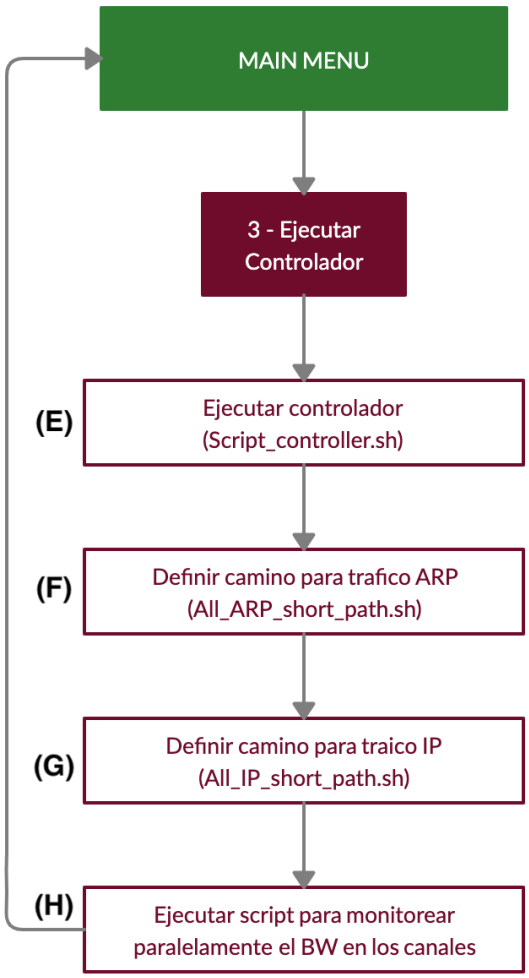



Figura 4-12 – Diagrama. OPCION 3) Ejecutar Controlador

El script *Monitor.py* (caja H) de la *Figura 4-12* es el ejecutor principal de esta solución. Una vez que es inicializado, éste permanece funcionando de forma paralela al controlador pero consumiendo datos que va recopilando y guardando de forma periódica en un archivo .scv tal cual se explicó en el punto 4.2.1. Cada 10 segundos el *Traffic Monitor* reescribe el archivo con los valores de anchos de banda libre de cada

enlace. Estos a su vez son copiados por el *Monitor*.py a dos archivos de texto llamados *Data_long.sh* y *Data.short.sh*, los cuales van guardando el dato históricos del enlace libre disponible. A su vez, el programa calcula el promedio porcentual de enlace disponible en los últimos 30 segundos y es a partir de este dato donde luego decide si es necesario aplicar políticas para el tratamiento del tráfico sensible. Si en los últimos 30 segundos se aprecia un uso del canal **superior al 65%**, entonces se ejecutan las políticas de tráfico que consisten en tres puntos:

1) Ejecutar un script que envía el tráfico sensible por el camino corto (short_path)

2) Ejecutar un script que envía el tráfico ARP por el camino largo (long_path)

3) Ejecutar un script que envía el tráfico IPv4 no sensible por el camino largo (long_path)

Si en los últimos 30 segundos se aprecia que la disponibilidad del canal vuelve a estar por debajo del 65%, entonces se ejecutan otros tres scripts que borran estas configuraciones recientemente mencionadas. De esta forma todo el tráfico IP, sea sensible como no, y el tráfico ARP viajan por el camino corto. Toda esta secuencia se puede ver mejor en el diagrama de al *Figura 4.13*.

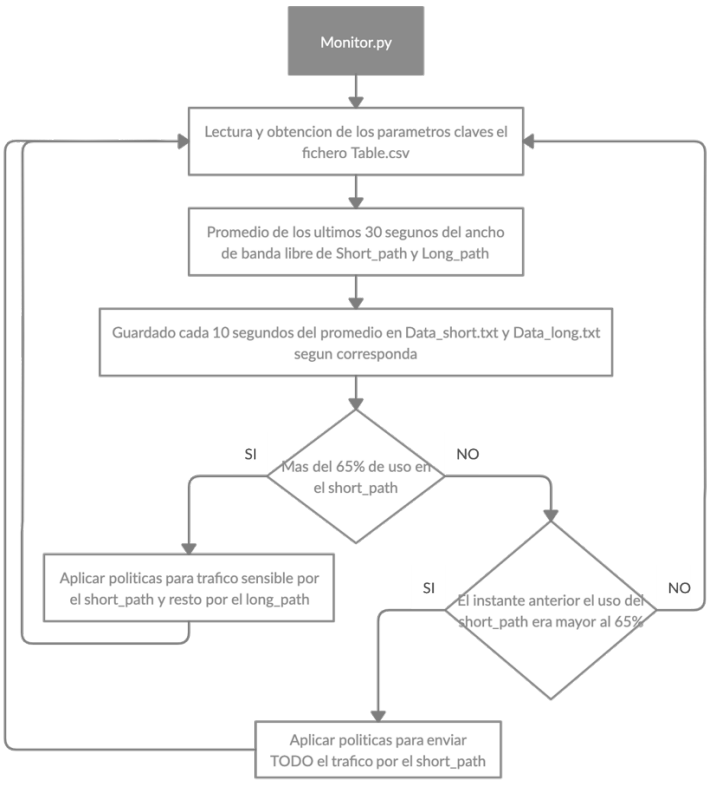



**Figura 4-13 – Diagrama Monitor.py**

Con esto se logra llegar a una solución permite que mientras el canal formado por los conmutadores A,C y E esté relativamente libre (menos del umbral crítico) todo el tráfico viaja por él. Sin embargo, cuando este supere el 65% de uso, todo el tráfico que no requiera mayores prestaciones se envía por el camino mas largo, dejando el corto solo para el tráfico sensible en caso de que haya. Esto en definitiva hace que el tráfico que requiera mayores prestaciones vaya siempre por el mejor camino, dejando el resto que vaya por el que le corresponda según el estado instantáneo de la red.

El valor del umbral fue elegido de forma arbitraria para la muestra de la solución pero es fácilmente modificable desde la variable "bw_lim" del *Monitor.py*.

El programa de monitoreo de enlaces se encuentra constantemente imprimiendo en una consola el porcentaje de enlace libre como muestra la *Figura 4-14*. De esta forma el operario puede conocer su estado en tiempo real, aunque también lo podrá ver en forma de gráfica como se muestra en el punto 4.4.5.

```
LONG_PATH (CONM_A - CONM_B - CONM_D - CONM_E):
    Bandwidth: 1.93 Kbps
    % libre (Prom. ultimos 30 seg.): 99.9981%
()
SHORT_PATH (CONM_A - CONM_C - CONM_E):
    Bandwidth: 1.93 Kbps
    % libre (Prom. ultimos 30 seg.): 99.9981%
```

Figura 4-14 – Monitor.py

### 4.4.4 OPCIÓN 4: Realizar configuración

La opción número cuatro permite al usuario realizar configuraciones de forma manual, fácil y rápida. Una vez que se accede, se despliega un menú que permite: Agregar configuración IP (1) , agregar configuración ARP (2) , eliminar configuración IP (3) tal cual se muestra en la *Figura 4-15*. A su vez todo el proceso que se explica a continuación se puede acompañar del diagrama de la *Figura 4-16*.

```
   ######  ########         ##     ##    ###    ##    ##
  ##    ## ##     ##        ##  ##  ##   ## ##   ###   ##
  ##       ##     ##        ##  ##  ##  ##   ##  #### ##
   ######  ##     ## ####### ##  ##  ## ##     ## ## ## ##
        ## ##     ##        ##  ##  ## ######### ##  ####
  ##    ## ##     ##        ##  ##  ## ##     ## ##   ###
   ######  ########          ###  ###  ##     ## ##    ##

============================================================
      DESARROLLO DE APLICACIONES DE INGENIERÍA DE TRÁFICO EN
              REDES WAN BASADAS EN SOFTWARE
============================================================
TFM de Andrés Jorge Muracciole Vázquez del Master en Ingeniería en
Redes y Servicios Telemáticos de la Universidad Politécnica de
Madrid (UPM)

----------------------------
        CONFIGURACIÓN
----------------------------
1) Agregar configuración IP
2) Agregr configuración ARP
3) Eliminar configuración IP

**Ingrese otra tecla para volver al menu principal**
```

Figura 4-15 – Menú de las posibles configuraciones

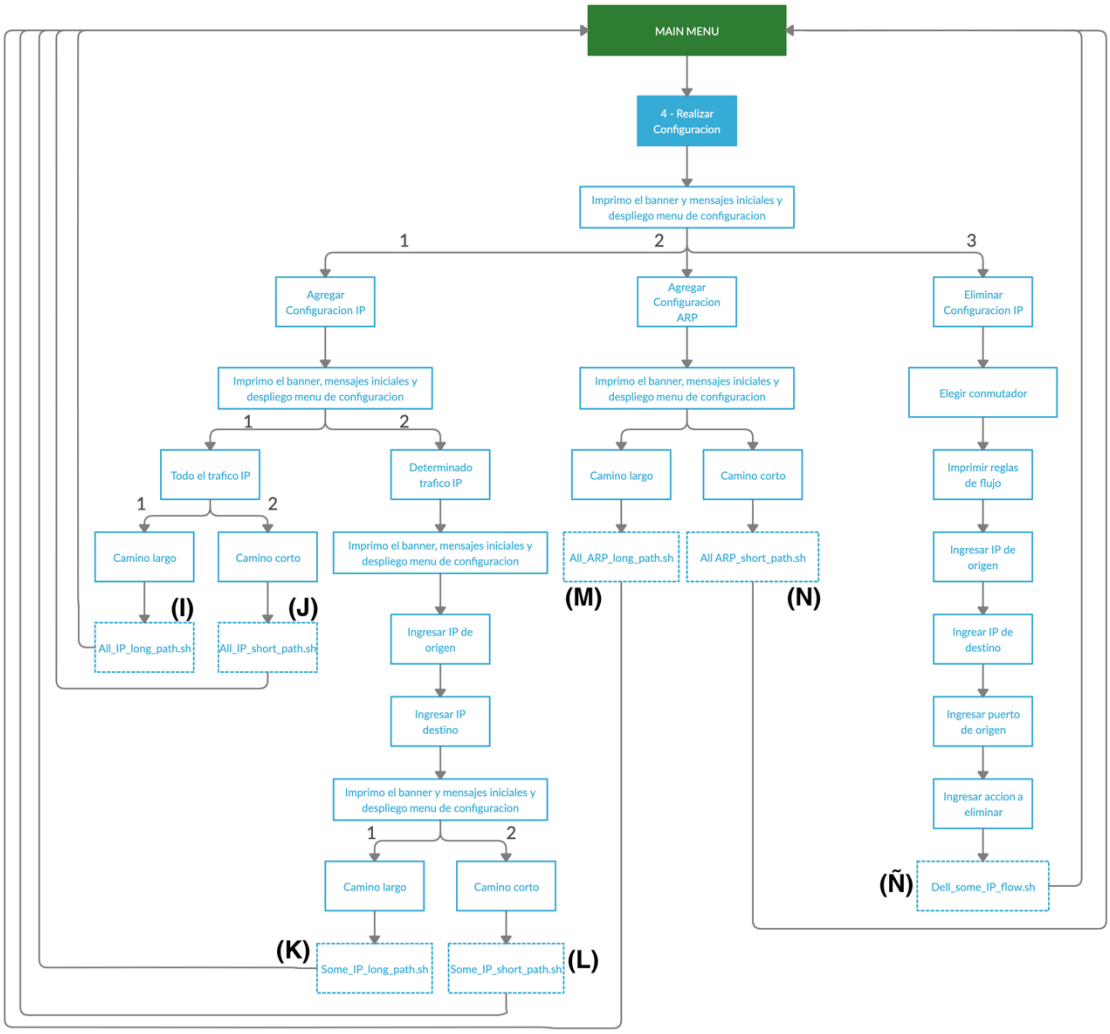



Figura 4-16 – Diagrama. OPCION 4) Realizar Configuración

### 1. *Agregar configuración IP*

El administrador es capaz de poder configurar de forma centralizada el camino que deben tomar todos los paquetes IPv4 de forma genérica (1) como también especificar un tráfico en particular a partir de la dirección de red del origen y del destinatario (2). A partir del tipo de configuración que se desee, es que el cliente debe marcar la opción. Introducir un número o carácter incorrecto le permite cancelar la operación y volver a la pantalla inicial.

En caso de optar la primer opción, luego es necesario elegir explícitamente el camino por el cual enviarlo, es decir, por el "long_path" (caja I) o el "short_path" (caja J). De tomar la opción 2, primero debe ingresar las direcciones IP de origen y destino. A modo de simplificar se asume que las redes son todas /24 por lo cual no es necesario especificar la máscara. Una vez introducida esa información se decide el camino (caja K o L) como se hizo anteriormente, quedando así configurada la ruta en los conmutadores para tratar este tráfico.

La configuración manual es prioritaria respecto a la configuración automática creada por el programa de Monitor.py. Esto hace que si hay una configuración realizada por el operario de forma manual, entonces siempre se va a cumplir sin importar si la solución TE intenta direccionar ese tráfico por otro camino. La forma de hacer esto es mediante el campo de prioridad de OpenFlow. Las reglas introducidas manualmente para cursar el tráfico a partir de una determinada dirección origen y destino se crean con **prioridad 10**, las entradas introducidas automáticamente por la solución de ingeniería de tráfico se configuran con **prioridad 5** y el tráfico configurado manualmente sin indicar origen y destino así como la configuración inicial donde determina que los paquetes IP y ARP al iniciar el programa vayan por el camino corto se ponen con **prioridad 0**.

#### *2. Agregar configuración ARP*

Aquí solo se permite elegir el camino que debe tomar todo el tráfico ARP (caja M o N) y no diferenciado como en el punto anterior. Esto se debe particularmente a que este tipo de paquetes son cursados entre los routers de borde (edge), por lo cual no tiene sentido diferenciar un determinado tipo de ARP ya que para este escenario siempre el origen y destino serán los únicos dos routers que existen. De forma análoga a lo anterior, estas entradas de flujo se programan con prioridad 1.

*Tanto para el tráfico IP como ARP solo es necesario configurar el camino de ida ya que la vuelta se configura automáticamente de forma análoga.*

#### *3. Eliminar configuración IP*

Retomando el punto donde se habla del agregado de configuración IP, se ve que deberá existir la forma de poder eliminar la configuración realizada. De no existir esta opción el sistema queda atado a que no haya cambios futuros ya que una vez que se crea la regla, ésta quedará con una prioridad tan alta que no habrá forma de que exista otra mas importante y por ende la configuración será permanente.

Una vez introducida la opción de eliminar la configuración, es necesario elegir el conmutador donde exista la regla a borrar. Una vez seleccionado, se despliegan las diversas entradas de la tabla de flujo para que el administrador escoja la que desea eliminar. La forma que tiene de hacerlo es introduciendo una serie de datos solicitados por el controlador (IP de origen, IP de destino, puerto de entrada y acción) los cuales se obtienen de la tabla. Una vez puesto los datos correctamente, la entrada deja de existir (caja Ñ).

*Aclaración: Solo es posible eliminar las entradas introducidas manualmente que cuenten con prioridad 10, es decir el tráfico con determinada IP de origen y destino. No así el resto.*

### 4.4.5 OPCIÓN 5: Visualizar configuraciones

A diferencia de la opción anterior donde el administrador realiza configuraciones en la red, aquí lo que se hace es permitir visualizar las mismas, las cuales van desde algo

muy sencillo como la versión del OpenvSwitch, hasta gráficas en tiempo real del tráfico cursado por ambos enlaces. El menú correspondiente a este punto se muestra en la *Figura 4-17* y se pasa a detallar cada una de las opciones en los siguientes apartados.

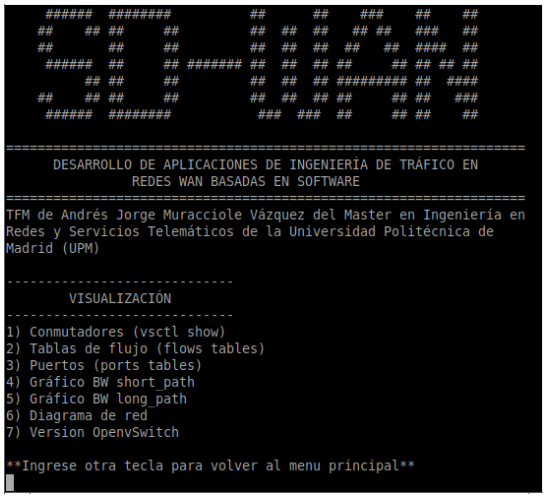



Figura 4-17 – Menú de las posibles visualizaciones

La *Figura 4-18* indica el diagrama de flujos correspondiente a todo lo referido a la visualización de configuraciones.

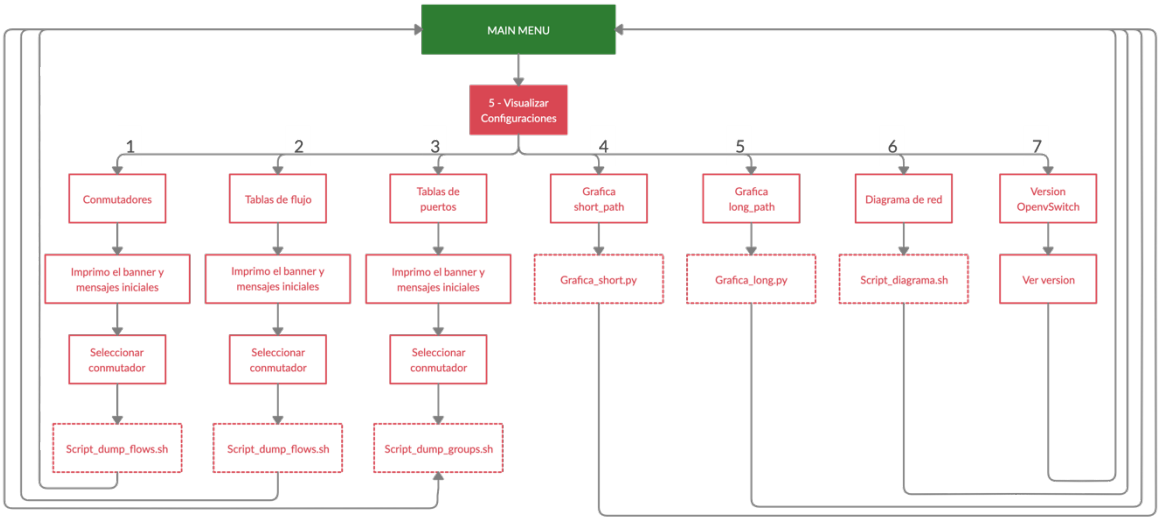



Figura 4-18 – Diagrama. OPCION 5) Visualizar Configuraciones

### 1. *Conmutadores (vsctl show)*

Aquí el administrador es capaz de obtener la información de los cinco conmutadores que forman la topología del escenario. Permite ver el nombre, el estado de la conexión con el controlador y las interfaces de los mismos.

### 2. *Tablas de flujo (flows tables)*

Mediante esta opción es posible ver las tablas de flujos de los conmutadores y así entender qué comportamiento están asumiendo cada uno a la hora de redirigir los

paquetes. A modo de ejemplo se muestra la *Figura 4-19* correspondiente a la tabla del CONM_A cuyos campos se detallan a continuación [viii].

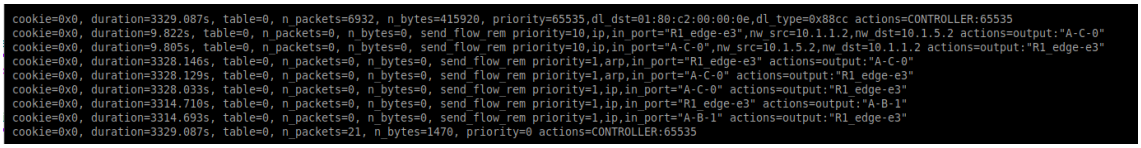



**Figura 4-17 – Tabla de flujos den CONM_A**

*cookie:* Campo identificador asociado al flow. No se tiene en cuenta en este trabajo por lo que vale 0 por defecto.

*duration:* Tiempo expresado en segundos que existe la entrada de flujo.

*table:* Identificador de la tabla de flujos. Puede que existan mas de una por cada conmutador.

*n_packets:* Cantidad de paquetes que fueron conmutados a partir de dicha entrada de flujo.

*n_bytes:* Cantidad de bytes que fueron conmutados a partir de dicha entrada de flujo.

*priority:* Valor de la prioridad. Ante dos posibles entradas coincidentes, el controlador elige conmutarlo a partir de la regla que tenga mayor prioridad. Este valor oscila entre 0 y 65535.

*in_port:* Nombre del puerto por el cual llegan los paquetes al conmutador.

*nw_src:* Dirección IPv4 de origen. Este campo puede o no estar presente. En caso de que no, se puede leer como "any".

*nw_dst:* Dirección IPv4 de destino. Este campo puede o no estar presente. En caso de que no, se puede leer como "any".

*actions:* Determina la acción a realizar con el tráfico que haya coincidido con los campos anteriores. Puede ir desde enviarlo al controlador, sacarlo por un determinado puerto o descartar el paquete.

### *3. Tabla de puertos (ports_tables)*

Permite ver información más detallada de los puertos de los conmutadores. A demás de la información obtenida de las entradas de flujo de la *Figura 4-17*, se puede obtener información mas detallada como la cantidad de paquetes descartados o con errores, entre otros.

#### *4 y 5. Gráfico BW short_path y gráfico BW long_path*

El administrador de la red tiene la capacidad de ver en tiempo real el tráfico (en Mbps) en función del tiempo para ambos caminos como se muestra la *Figura 4-18* y la *Figura 4-19.*

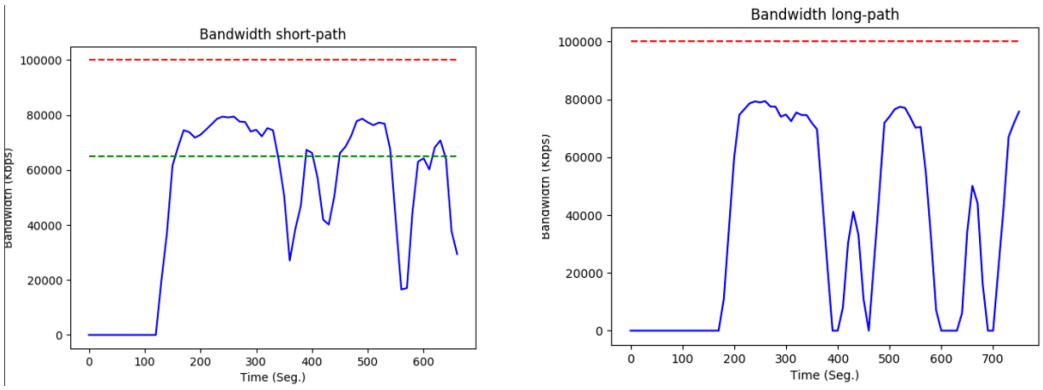



**Figura 4-18 (Izq.) – Gráfica Bandwidth (Kbps) en función del tiempo (seg.) de short_path**

**Figura 4-19 (der.) – Gráfica Bandwidth (Kbps) en función del tiempo (seg.) de long_path**

Estas gráficas surgen debido a que el programa *Monitor.py* exporta de forma diferenciada los datos obtenidos de ocupación del "long_path" y "short_path" a dos ficheros txt. Por otro lado, al seleccionar las opciones 4 o 5 dentro de la opción de "Visualizar configuraciones" estos ejecutan los scripts *Graph_short.py* (caja R) y *Graph_long.py* (caja S) respectivamente los cuales obtienen los valores de sus archivos de texto y los grafican en pantalla.

Las gráficas son interactivas con el usuario de forma que se permite hacer zoom, modificar sus tamaños o guardar una captura en su máquina local.

#### *6. Diagrama de red*

Desde la caja T es posible visualizar el diagrama de red del escenario en cuestión, devolviendo como resultado la *Figura 4-6* vista anteriormente.

#### *7. Versión OpenVSwitch*

La finalidad de esta opción es devolver la versión de los OVS´s que forman el escenario virtual.

### 4.4.6 OPCIÓN 6: Salir

La última opción del menú principal corresponde a "salir". Una vez elegida, esta borra las configuraciones del Wondershaper hechas en el punto 4.4.1, destruye el escenario tal cual se hace en el punto 4.4.2, deshabilita el controlador Ryu y cierra la ventanas con el escenario virtual levantado en el punto 4.4.5 en caso de estar abierto. De esta forma se asegura que no queda nada ejecutándose en segundo plano. Por último da por finalizado el programa haciendo desaparecer el menú principal.

## 4.5 Tráfico sensible

Llamaremos "tráfico sensible" a todo aquel que se requiera, por parte del administrador de la red, proporcionarle una prioridad respecto al resto. Con la ayuda del documento "SD-WAN Quality of service" de Aruba [x] se determinó que los siguientes protocolos serán los designados como sensibles para esta práctica:

| CATEGORÍA | PROTOCOLO | PUERTO TCP | PUERTO UDP |
|---|---|---|---|
| **Tiempo Real** | SIP | 5060 | 5060 |
| **Tiempo Real** | SSH | 22 | |
| **Túnel** | IPsec | - | 500 y 4500 |

Tabla 4-2 – Protocolos catalogados como tráfico sensible

Se tratarán todos los protocolos de la *Tabla 4-2* de igual forma a modo de simplificar el funcionamiento de la solución.

A la hora de hacer las pruebas del apartado 5 se utiliza la herramienta iperf. Esta no permite utilizar puertos reservados con lo cual se hace el mapeo de la *Table 4-3* para simular estos tráficos.

| PUERTO TCP REAL | PUERTO TCP FICTICIO | PUERTO UDP REAL | PUERTO UDP FICTICIO |
|---|---|---|---|
| 5060 | 35000 | 5060 | 35000 |
| 22 | 35001 | 500 y 4500 | 35001 y 35002 |

Tabla 4-3 – Mapeo de puertos para utilizar en iperf

# 5 Pruebas y resultados obtenidos

## 5.1 Experimento 1: Modificación de las tablas de flujo por medio del menú (manualmente)

En este experimento se buscará mostrar por un lado lo rápido e intuitivo que resulta modificar por medio de la interfaz el camino que toma el tráfico para ir de un origen a un destino. A su vez se estudiará la convergencia de hacer esto viendo las gráficas de tráfico por cada enlace.

En primer lugar se levanta el escenario y el controlador mediante las opciones 1 y 3 del menú. Viendo las tablas de flujo de CONM_A se aprecia en la línea 4 de la *Figura 5-1* que el tráfico IP proveniente el router R1 se conmuta hacia el CONM_C (A-C-0), es decir por le short_path.

```
1   cookie=0x0, duration=23.532s, table=0, n_packets=52, n_bytes=3120,
priority=65535,dl_dst=01:80:c2:00:00:0e,dl_type=0x88cc actions=CONTROLLER:65535

2   cookie=0x0, duration=18.785s, table=0, n_packets=0, n_bytes=0, send_flow_rem priority=1,arp,in_port="R1_edge-e3"
actions=output:"A-C-0"

3   cookie=0x0, duration=18.770s, table=0, n_packets=0, n_bytes=0, send_flow_rem priority=1,arp,in_port="A-C-0"
actions=output:"R1_edge-e3"

4   cookie=0x0, duration=18.696s, table=0, n_packets=0, n_bytes=0, send_flow_rem priority=1,ip,in_port="R1_edge-e3"
actions=output:"A-C-0"

5   cookie=0x0, duration=18.680s, table=0, n_packets=0, n_bytes=0, send_flow_rem priority=1,ip,in_port="A-C-0"
actions=output:"R1_edge-e3"

6   cookie=0x0, duration=23.537s, table=0, n_packets=3, n_bytes=210, priority=0 actions=CONTROLLER:65535
```

**Figura 5-1 – Tabla de flujo del CONM_A por defecto**

Haciendo un TCP iperf[11] de 50 Mbps entre h01 y h03 se ve efectivamente mediante las gráficas de la *Figura 5-2* y *Figura 5-3* que todo el tráfico va por el short_path.

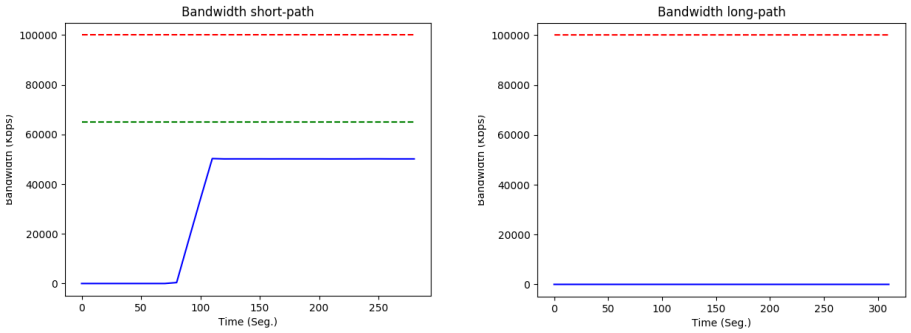



**Figura 5-2 (Izq.) – Ancho de banda utilizado del short_path en función del tiempo**

**Figura 5-3 (Der.) – Ancho de banda utilizado del long_path en función del tiempo**

Sin detener el iperf se hace un cambio de rutas a nivel el controlador [12] de forma que ahora todo el tráfico IP se encamine por el long_path. Esto se comprueba viendo tanto la línea 5 de la *Figura 5-4* la cual indica que el tráfico proveniente del router R1 se encamine hacia el CONM_B (A-B-1), como también las gráficas de tráfico de la *Figura 5-5* y *Figura 5-6*.

[11] **Servidor**: "iperf -s -i 1 -t 1000 -b 50000000". **Cliente**: "iperf -c 10.1.5.2 -i 1 -t 1000 -b 50000000"
[12] Comandos: 4-1-1-1

```
1    cookie=0x0, duration=361.709s, table=0, n_packets=795, n_bytes=47700,
priority=65535,dl_dst=01:80:c2:00:00:0e,dl_type=0x88cc actions=CONTROLLER:65535

2    cookie=0x0, duration=358.789s, table=0, n_packets=13, n_bytes=546, send_flow_rem
priority=1,arp,in_port="R1_edge-e3" actions=output:"A-C-0"

3    cookie=0x0, duration=358.774s, table=0, n_packets=13, n_bytes=546, send_flow_rem priority=1,arp,in_port="A-C-0"
actions=output:"R1_edge-e3"

4    cookie=0x0, duration=358.679s, table=0, n_packets=39563, n_bytes=2611226, send_flow_rem
priority=1,ip,in_port="A-C-0" actions=output:"R1_edge-e3"

5    cookie=0x0, duration=23.388s, table=0, n_packets=55773, n_bytes=1829730266, send_flow_rem
priority=1,ip,in_port="R1_edge-e3" actions=output:"A-B-1"

6    cookie=0x0, duration=23.370s, table=0, n_packets=14961, n_bytes=1018718, send_flow_rem priority=1,ip,in_port="A-
B-1" actions=output:"R1_edge-e3"

7    cookie=0x0, duration=361.709s, table=0, n_packets=12, n_bytes=840, priority=0 actions=CONTROLLER:65535
```

Figura 5-4 – Tabla de flujo del CONM_A luego de cambiar la configuración de ruteo

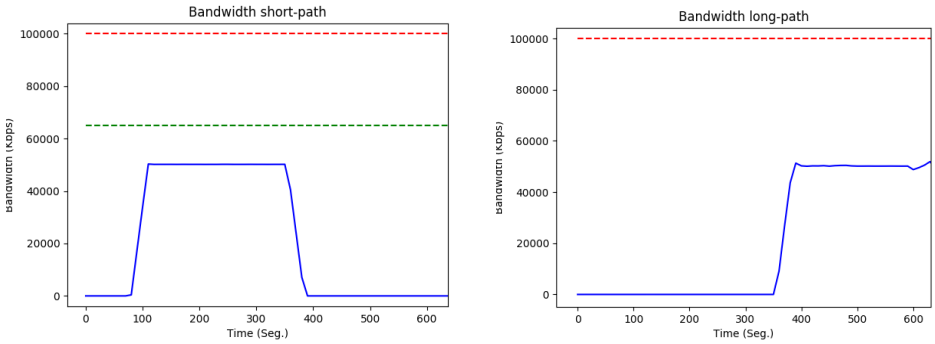



Figura 5-5 (Izq.) – Exp. 1. Ancho de banda utilizado del short_path en función del tiempo después de la configuración de caminos

Figura 5-6 (Der.) – Exp. 1. Ancho de banda utilizado del long_path en función del tiempo después de la configuración de caminos

De esta forma se puede apreciar que el cambio en los conmutadores se hace de forma correcta e instantánea. Por otra parte se desprende la simpleza con la es efectuado el cambio de configuración y lo escalable que es respecto a una solución tradicional ya que en ese caso hubiera sido necesario entrar a todos los equipos uno por uno y modificar las reglas lo cual trae un caída en el servicio.

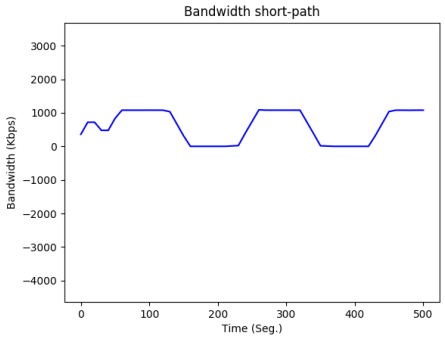
## 5.2 Experimento 2: Disponibilidad del servicio ante cambios de configuración

Aquí se busca determinar si ante un cambio repentino del camino por el cual se cursa el tráfico, existen cortes de servicio o paquetes caídos. Para ello se ejecuta un iperf [13] UDP durante 500 segundos desde el hosts h01 hacia h03 mientras se va variando cada 100 segundos el camino por el cual viajan los paquetes como se aprecia en la *Figura 5-7* y en la *Figura 5-8*. Una vez transcurrido el tiempo se obtienen los siguientes resultados:

- Paquetes enviados: 44583

[13] **Servidor**: "iperf -s -i 1 -t 500 -u". **Cliente**: "iperf -c 10.1.5.2 -i 1 -t 500 -u"

- Paquetes perdidos: 10 (0,022%)
- Jitter promedio: 0,044 ms



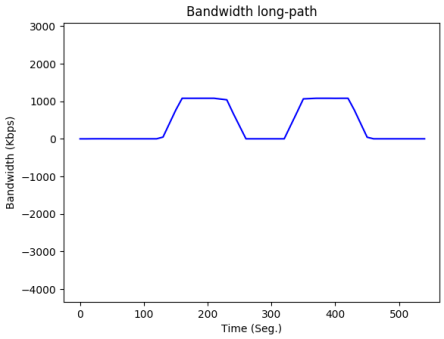




**Figura 5-7 (Izq.) – Exp. 2. Ancho de banda utilizado del short_path en función del tiempo**

**Figura 5-8 (Der.) – Exp. 2. Ancho de banda utilizado del long_path en función del tiempo**

Se hizo captura de pantalla cada 100 de los resultado del iperf, obteniendo los siguientes datos:

| Interval | Transfer | Bandwidth | Jitter | Lost/Total Datagrams |
|---|---|---|---|---|
| 100.0-101.0 sec | 113 KBytes | 929 Kbits/sec | 0.037 ms | 10/ 89 (11%) |
| 200.0-201.0 sec | 126 KBytes | 1.03 Mbits/sec | 0.035 ms | 0/ 88 (0%) |
| 300.0-301.0 sec | 129 KBytes | 1.06 Mbits/sec | 0.032 ms | 0/ 90 (0%) |
| 400.0-401.0 sec | 126 KBytes | 1.03 Mbits/sec | 0.033 ms | 0/ 88 (0%) |

De esta forma se puede ver que todos los paquetes perdidos fueron resultado del primer cambio de configuración, pero en las siguientes, la pérdida de paquetes fue de 0%. El motivo por le cual sucedió esto es que en el primer cambio fue necesario crear una entrada nueva en los conmutadores pero en las demás lo único que se hizo fue una modificación de las mismas con lo cual la convergencia fue inmediata. Esto se puede apreciar también si se comparan la *Figura 5-1* y la *Figura 5-4*. En esta última hay una entrada extra (línea 6) mientras que la entrada de la línea 5 es la modificada.

Estos resultados refuerzan la idea de una solución robusta y de convergencia inmediata ante cambios bruscos de encaminamiento.

## 5.3 Experimento 3: Encaminamiento de tráfico sensible SIN solución TE

Los experimentos 3 y 4 buscan mediante la práctica, mostrar la importancia de una solución de ingeniería de tráfico de este estilo. Las pruebas serán las mismas con la diferencia que para este caso se deshabilitará el programa *Monitor.py*, con lo cual no se hará re direccionamiento del tráfico sensible.

Mediante la comparación de resultados se busca poder determinar las ventajas que se obtienen aplicando TE en cuanto a la pérdida de paquetes, la capacidad de transmisión y las velocidades obtenidas.

El experimento consiste en simular un "tráfico sensible" UDP (por ejemplo voz sobre IP) a 60 Mbps entre h01 y h03 [14] y un tráfico no sensible TCP también de 60 Mbps entre h02 y h04 [15]. En esta prueba se deshabilita el TE de forma que se estudia cómo competirán ambos por el canal.

Pasados los 300 segundos configurados en el iperf se obtienen los siguientes resultados.

1) El tráfico siembre fue por el camino corto y nada por le largo tal cual se puede ver en la *Figura 5-9* y la *Figura 5-10* respectivamente.

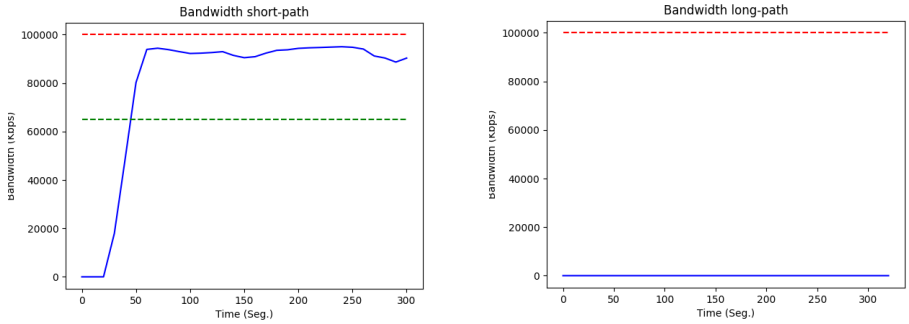



**Figura 5-9 (Izq.) – Exp. 3. Ancho de banda utilizado del short_path en función del tiempo sin TE**

**Figura 5-10 (Der.) – Exp. 3. Ancho de banda utilizado del long_path en función del tiempo sin TE**

2) No solo no se tuvo en cuanta la categorización de tráfico sensible sino que al ser del tipo UDP frente a uno TCP, este tuvo pérdidas ya que el tráfico sumado entre ambos era de 120 Mbps pero estos estaban compitiendo en un canal de 100 Mbps. Mas precisamente el "tráfico sensible" tuvo unas pérdidas del 24% (370641 paquetes en un total de 1530492).

En definitiva se ve que no hubo un uso óptimo de la red ya que no se aplicaron políticas de servicio, hubo una importante cantidad de pérdida de paquetes habiendo un camino de 100 Mbps sin utilizar.

## 5.4 Experimento 4: Encaminamiento de tráfico sensible CON solución TE

Para este caso se realiza la misma prueba que en el experimento 3, a excepción de que ahora si se activa la solución de ingeniería de tráfico, con lo cual a medida que el programa detecte que se supera el umbral de 65% del uso del canal, este realizará configuraciones en el controlador para intentar optimizar el tráfico. De esta forma dejará el tráfico UDP en el camino corto, enviando el resto no sensible por el camino largo.

[14] **Servidor**: "iperf -s -i 1 -t 300 -p 35000 -u -b 60000000". **Cliente**: "iperf -c 10.1.5.2 -i 1 -t 300 -p 35000 -u -b 60000000"
[15] **Servidor**: "iperf -s -i 1 -t 300 -b 60000000". **Cliente**: "iperf -c 10.1.5.3 -i 1 -t 300 -b 60000000"

Una vez hecho esto se obtuvo las gráficas de la *Figura 5-11* y la *Figura 5-12*, del short_path y long_path respectivamente. Aquí se puede ver a diferencia del experimento anterior que:

1) Se envió tráfico por ambos enlaces, con lo cual se utilizó mejor la infraestructura de red existente.

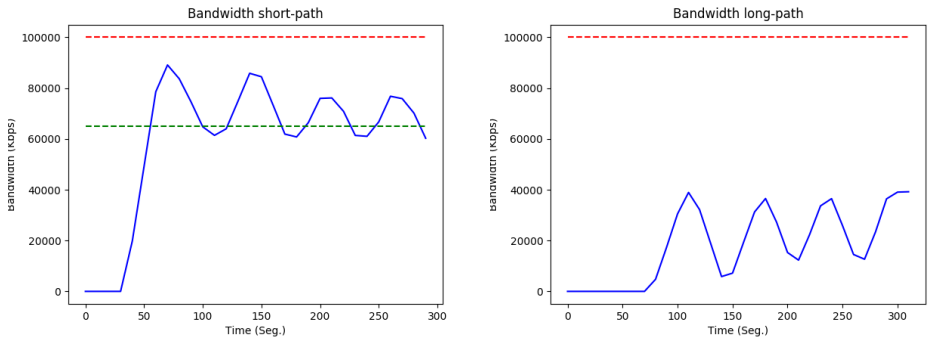



**Figura 5-11 (Izq.) – Exp. 4. Ancho de banda utilizado del short_path en función del tiempo con TE**

**Figura 5-12 (Der.) – Exp. 4. Ancho de banda utilizado del long_path en función del tiempo con TE**

2) Cuando se superaba el umbral crítico, el programa lo detectaba y aplicaba las políticas de tráfico sensible enviando todo aquel que no lo fuera por el camino mas largo.

3) En este caso las pérdidas de tráfico UDP promedio durante los 300 segundos fue de 16% (252162 paquetes de 1530514). Si bien la pérdida disminuyó, esta sigue siendo considerable. Esto se debe a que mientras competían por el mismo enlace las pérdidas eran de entre 20 y 30%. En el segundo en que se hacia el cambio de configuración en los conmutadores las pérdidas crecían considerablemente pero cuando el tráfico sensible disponía de todo el enlace, las pérdidas disminuían a menos de 8% como se puede ver en la secuencia de la *Figura 5-13*.

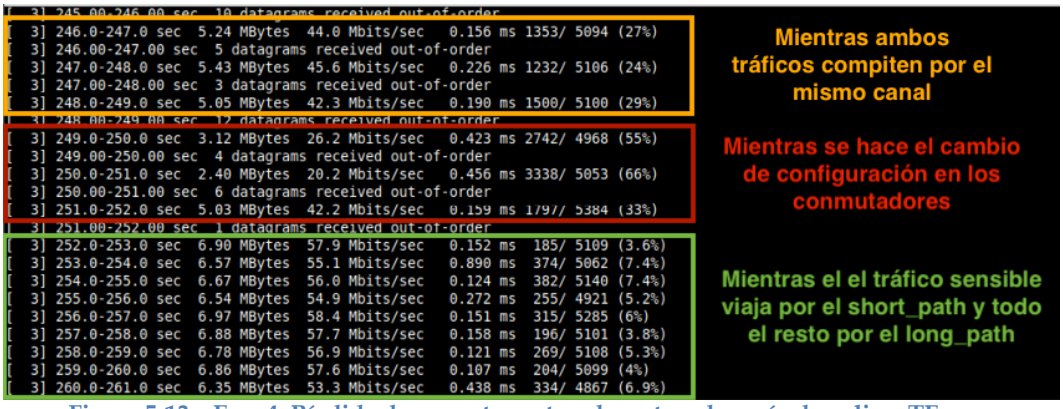



**Figura 5-13 – Exp. 4. Pérdida de paquetes entes, durante y después de aplicar TE**

4) De la *Figura 5-13* se desprende también que mientras compartían canal (sección amarilla) el bandwidth era un poco menor a cuando no lo hacían (sección verde). Con lo cual se puede decir que este tipo de ingeniería de tráfico permite optimizar los canales alcanzando así mejores velocidades de transmisión y recepción.

## 5.5 Experimento 5: Encaminamiento de tráfico sensible de gran tamaño CON solución TE

Esta ultima prueba es muy similar a los experimentos 3 y 4, con la diferencia de que ahora se simulará una prueba con un tráfico sensible muy pesado (100 Mbps) el cual supere el umbral crítico durante todo el intervalo que dure la prueba. Con esto se busca comparar las pérdidas de paquetes obtenida sin TE y luego de aplicar ingeniería de tráfico y ver si ante estos casos de estrés la mejoría es considerable.

Deshabilitando la TE, se simula un tráfico sensible UDP [16] entre h01 y h03 de 100 Mbps y un tráfico no sensible TCP [17] también de 100 Mbps entre h02 y h04. Una vez finalizada esa prueba se vuelve a realizar la simulación de estrés pero ahora con ingeniería de tráfico habilitada. En el cuadro de la *Tabla 5-1* se plasman los resultados.

| | **Sin TE** | **Con TE** |
|---|---|---|
| **Paquetes enviados** | 2550857 | 2550868 |
| **Paquetes perdidos** | 1361629 | 590558 |
| **Pérdida de paquetes (%)** | 53 | 23 |
| **Jitter promedio (ms)** | 0,171 | 0,056 |
| **Bandwidth UDP promedio (Mbps)** | 46,6 | 76,8 |

**Tabla 5-1 – Comparativa de datos obtenidos con y sin TE ante pruebas de estrés**

Aquí de puede notar aun más las mejorías que se obtienen aplicando ingeniería de tráfico. De esta tabla se desprende que:

- Las pérdidas se redujeron a mas de la mitad
- El jitter promedio disminuyó a un tercio
- El bandwidth aumentó un 60% aproximadamente

[16] **Servidor**: "iperf -s -i 1 -t 300 -p 35000 -u -b 100000000". **Cliente**: "iperf -c 10.1.5.2 -i 1 -t 300 -p 35000 -u -b 100000000"
[17] **Servidor**: "iperf -s -i 1 -t 300 -b 100000000". **Cliente**: "iperf -c 10.1.5.3 -i 1 -t 300 -b 100000000"

# 6 Conclusiones y líneas futuras

## 6.1 Conclusiones

Una vez finalizadas las pruebas y obtenido los resultados plasmados en el capítulo 5 de esta documento, se pueden sacar conclusiones al respecto.

En cuanto al alcance del trabajo, se concluye que se llegó a cumplir con creces el objetivo planteado en un principio. Se realizó el diseño e implementación de una solución TE acorde a las necesidades, la cual fuera capaz de medir el estado de los enlaces en tiempo real y a partir de dicha información elegir el mejor camino por el cual encaminar los paquetes. Si bien en un principio se había pensado que el criterio para definir el QoS dependería del ancho de banda, pérdida de paquetes y delay, al final se vio necesario reducir la complejidad optando únicamente por tomar como parámetro el bandwidth libre.

Una segunda conclusión a destacar es que se logró ir un poco más allá del objetivo planteado al iniciar el trabajo. Si bien idea principal era hacer el desarrollo e implementación de una solución TE, a esta se le agregó todo lo referido a la interfaz gráfica para su administración. De esta forma se vio que resulta ser extremadamente útil contar con ella para que el administrador de la red tenga la libertad de poder realizar mas configuraciones y potenciar aun mas el alcance que le puede dar a su red.

Respecto a la complejidad del proyecto, éste contó con una dificultad acorde para lo que es una tesis de fin de master en ingeniería. Para su desarrollo fue necesario fortalecer conocimientos de redes SDN, así como también de programación Shell y Python. Por otra parte fue inevitable conocer como integrar ambas tecnologías para que se complemente una con la otra.

En cuanto a las conclusiones técnicas, se afirma que efectivamente una solución como la planteada en este trabajo permite optimizar las redes con topologías multicamino, obteniendo mejoras considerables como: **usos mas eficiente de los enlaces, menor tasa de pérdida de paquetes, mayores tasas de velocidades alcanzadas y menores tiempos de jitter**. Por otra parte se concluye que uno de los puntos mas destacables es la **capacidad de escalabilidad** que tiene una solución de este tipo, ya que permite aplicar una capa de programación a la red, de forma de poder alcanzar los requisitos deseados. Esto es impensable en redes tradicionales ya que ante la ausencia de la figura del controlador, sería necesario realizar configuraciones de forma manual en todos los componentes de la red en caso de querer modificar una ruta. Queda en evidencia que a mayor cantidad de saltos, mayor la complejidad y el tiempo que se requeriría plasmar estos cambios. Con SDN no solo es posible sino que permite hacer "cambios en caliente" y en tiempo real o ventanas de mantenimiento extremadamente cortas sin ver casi afectado el servicio.

En definitiva este trabajo logró mostrar una posible solución de ingeniería de tráfico para encaminar y catalogar tráfico sensible sobre una red SD-WAN. A su vez permitió mostrar el funcionamiento de las tecnologías SDN y ver su potencial mediante el estudio de los resultados de varios experimentos.

## 6.2 Líneas futuras

Si bien la solución elegida finalmente para la decidir el mejor camino que debe tomar un paquete fue midiendo el ancho de banda disponible, cuanto más información se tanga para la elección de esto, es mejor. Por tal motivo es que la primer línea futura que se plantea es realizar una optimización de esta solución de TE, donde el criterio de selección sea a partir de mas de un factor (delay, pérdida de paquetes, métricas, etc.)

En segundo lugar se ve con buenos ojos aplicar un trabajo de frontend para implementar de una interfaz gráfica aún mas amigable a la ya existente con el usuario. Esto implicaría por ejemplo que el operario pueda seleccionar opciones y no solo introducir la información mediante la interfaz de la consola. En definitiva uno de los objetivos por el cual se planteó realizar una interfaz fue para ayudar al administrador y hacer mas rápido e intuitivo el trabajo, con lo cual mejorando la interfaz esto sería aun mas práctico.

Una tercer y última línea que queda como propuesta a mejorar es el proceso referido a "tráfico sensible" donde hoy este es configurado manualmente desde el script *Monitor.py*. Se ve como buena idea darle la posibilidad al usuario de que este pueda seleccionar manualmente por medio del menú principal qué tipo de tráfico deberá ser catalogado de esta forma. Por otra parte, por como está implementada la solución, ésta solo diferencia el tráfico normal del sensible, pero dentro de este último podrían existir diversas clasificaciones, como suele suceder en las implementaciones reales. Se plantea la posibilidad de adaptar el código ya hecho de forma que dentro de los tráficos sensibles, unos as su vez sean más prioritarios que otros.

# Bibliografía


**[i]** Nuage Networks, *Virtualized Network Services*. [Online] de https://onestore.nokia.com/asset/183178

**[ii]** Cisco, *Cisco SD-WAN Solution Overview*. [Online] de https://www.cisco.com/c/en/us/solutions/collateral/enterprise-networks/sd-wan/nb-06-sd-wan-sol-overview-cte-en.html

**[iii]** FlexiWAN, B4: Experience with a Globally-Deployed Software Defined WAN. [Online] de https://docs.flexiwan.com/

**[iv]** UCSD, B4: Expericnce with a Globally-Deployed Software Defined WAN. [Online] de https://cseweb.ucsd.edu/~vahdat/papers/b4-sigcomm13.pdf

**[v]** Logicalis, Software Defined Networks: el future de las arquitecturas de red. [Online] de https://www.la.logicalis.com/globalassets/latin-america/logicalisnow/revista-20/lnow20-nota-42-45.pdf

**[vi]** Sdxcentral, What is Ryu Controller?. [Online] de https://www.sdxcentral.com/networking/sdn/definitions/what-is-ryu-controller/

**[vii]** ResearchGate, Controladores SDN, elementos para su selección y evaluación. [Online] de https://www.researchgate.net/publication/320711755_Controladores_SDN_elementos_para_su_seleccion_y_evaluacion

**[viii]** Ryu, ryu.app.ofctl_rest. [Online] de https://ryu.readthedocs.io/en/latest/app/ofctl_rest.html

**[ix]** RoutingFreak!, The Classical Fish Problem in Routing. [Online] de https://routingfreak.wordpress.com/2008/03/09/the-classical-fish-problem/

**[x]** Aruba, SD-WAN Quality of service. Supplemental Guide.