

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**

Tutor  
**Carlos M. 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 y el tráfico es cada vez mayor. Sin embargo, las redes que transportan los datos no han sufrido grandes cambios desde sus orígenes.

Como forma de acompañar dicho crecimiento y previendo la demanda a futuro surge la necesidad de aplicar métodos para flexibilizar las redes. A raíz de ello nacen las *redes definidas por software*, las cuales permiten facilitar la implementación de servicios de una manera mucho mas dinámica, escalable y centralizada. Estas redes desacoplan el plano de datos del de control, con lo cual la gestión y administración de la red se ve ampliamente beneficiada ya que facilita la implementación de nuevas técnicas como son las diversas soluciones de ingenierías de tráfico.

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 son algunas e las posibilidades que ofrece SD-WAN.

Este trabajo busca implementar una solución de ingeniería de tráfico que permita aplicar calidad de servicio sobre tráfico WAN multicamino, seleccionando en tiempo real el camino óptimo que deben tomar los paquetes. Para ello se diseñó una solución mediante la cual se monitorean los enlaces de la red WAN de un escenario virtual desarrollado con la herramienta VNX por medio de consultas a la API del controlador Ryu. Ryu permite modificar las tablas de flujo de los comutadores en tiempo real. De esta forma se van creando las reglas de flujo en los comutadores para que estos tomen el mejor camino en función de los requisitos de calidad de servicio de las aplicaciones.

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 con respecto a las garantías de calidad de servicios de las aplicaciones: más ancho de banda, menores pérdidas de paquetes y menores retardos.



## Abstract

We are living in an age where technologies are constantly evolving. Every day new ideas, applications and services emerge searching to solve the needs of users, seeking to achieve the best quality parameters and making the experience more instantaneous and real. Platforms migrate to the cloud and traffic is increasing. However, networks have not suffered major changes since their origins.

The way to accompany this growth and anticipating the demand for future increase is necessary to apply methods to make the networks more flexible. As a result, software-defined networks are born, which allow for the implementation of services in a more dynamic, scalable and centralized way. These networks decouple the data plane of the control plane. This helps management and administration tasks and facilitates the implementation of new techniques such as the various traffic engineering solutions.

Using methods to modify traffic handling on WAN networks is not something new. MPLS solutions are the primary implementation in traditional networks that provide for a quality of service. These tend to be expensive and not very manageable. The adaptation of the SDN concept to WAN networks opens a new paradigm of how to carry out these processes. Links optimizations, balance traffic or routing decisions in real time based on the state of the network are some of the possibilities that SD-WAN offers.

This project aims to implement a traffic engineering solution that allows for quality of service on multi-path WAN traffic, selecting in real time the path that packets should be take. The solution has been designed through the quality of the WAN network links, which are monitored in a virtual scenario developed with the VNX tool through queries to the Ryu controller API. Ryu allows to modify the switch flow tables in real time. In this way, flow rules are created in switches so that they take the best path depending on the quality of service requirements of the applications.

This solution brings a significant improvement in scalability and centralization of network administration. Additionally, positive results have been obtained concerning the guarantees of quality of service of the applications: more bandwidth, less packet loss and less delay.



# Índice general

Resumen.....	i
Abstract .....	iii
Índice general .....	v
Índice de figuras .....	ix
Índice de tablas .....	xi
Índice de códigos y ecuaciones.....	xii
Siglas.....	xiii
1 Introducción .....	1
2 Estado del arte.....	4
2.1 Soluciones comerciales académicas .....	7
2.1.1 Nuage .....	7
2.1.2 Viptela .....	9
2.1.3 FlexiWAN .....	10
2.2 Otros trabajos de investigación con ingeniería de tráfico SDN .....	11
2.2.1 B4.....	12
3 Casos de uso .....	13
3.1 Caso de uso A.....	13
3.2 Caso de uso B.....	14
4 Aplicación de ingeniería de tráfico SD-WAN .....	16
4.1 Pruebas preliminares sobre servicios SD-WAN de código abierto .....	16
4.2 Escenario virtual .....	18
4.2.1 Wondershaper.....	22
4.3 Controlador Ryu .....	22
4.3.1 Traffic Monitor.....	23
4.3.2 Ofctl_rest.....	25
4.4 Desarrollo de la aplicación SDN para ingeniería de tráfico .....	26
4.4.1 OPCION 1: Crear escenario .....	27

4.4.2	OPCIÓN 2: Destruir escenario .....	28
4.4.3	OPCIÓN 3: Ejecutar controlador.....	29
4.4.4	OPCIÓN 4: Realizar configuración.....	33
1.	Agregar configuración IP .....	34
2.	Agregar configuración ARP.....	35
3.	Eliminar configuración IP .....	35
4.4.5	OPCIÓN 5: Visualizar configuraciones.....	36
1.	Conmutadores (vsctl show) .....	37
2.	Tablas de flujo (flows tables) .....	37
3.	Tabla de puertos (ports_tables) .....	38
4 y 5.	Gráfico BW short_path y gráfico BW long_path .....	39
6.	Diagrama de red .....	39
7.	Versión OpenvSwitch .....	40
4.4.6	OPCIÓN 6: Salir.....	40
4.5	Tráfico sensible.....	40
5	Pruebas y resultados obtenidos .....	41
5.1	Experimento 1: Modificación de las tablas de flujo por medio del menú (manualmente) .....	41
5.2	Experimento 2: Disponibilidad del servicio ante cambios de configuración	
	43	
5.3	Experimento 3: Encaminamiento de tráfico sensible SIN solución TE .....	45
5.4	Experimento 4: Encaminamiento de tráfico sensible mediante soluciones de TE	
	46	
5.5	Experimento 5: Encaminamiento de tráfico sensible de gran tamaño SIN y CON solución TE .....	47
6	Conclusiones .....	49
6.1	Conclusiones.....	49
6.2	Líneas futuras .....	50
Bibliografía .....		51
ANEXO A: ASPECTOS ÉTICOS, ECONÓMICOS, SOCIALES Y AMBIENTALES..		53
A.1 INTRODUCCIÓN .....		53
A.2 DESCRIPCIÓN DE IMPACTOS RELEVANTES RELACIONADOS CON EL PROYECTO .....		53

A.3 ANÁLISIS DETALLADO DE ALGUNO DE LOS PRINCIPALES IMPACTOS	54
.....	
A.4 CONCLUSIONES.....	54



## Índice de figuras

Figura 1 – Arquitectura SDN [1] .....	5
Figura 2 – Arquitectura MEF [2] .....	6
Figura 3 – Red sin interrupciones SD-WAN [3].....	8
Figura 4 – Arquitectura Nuage [3] .....	9
Figura 5 – Arquitectura Viptela [4] .....	10
Figura 6 – Diagrama de bloques de alto nivel de la arquitectura FlexiWAN [5].....	11
Figura 7 – Escenario modelo de caso de uso A .....	14
Figura 8 – Escenario modelo de caso de uso 2 .....	15
Figura 9 – Visualización el FlexiEdge desde el FlexiManage.....	17
Figura 10 – Estadísticas de bps y pps en un FlexiWAN-router .....	18
Figura 11 – Escenario virtual con red de conmutadores en topología de pez.....	19
Figura 12 – Topología de pez.....	21
Figura 13 – Topología de pez.....	21
Figura 14 – Tablas de topología y puertos por tráfico del programa Traffic Monitor de Ryu.....	24
Figura 15 – Captura de tráfico ante solicitud de modificación de entrada de flujo hecha con Wireshark .....	25
Figura 16 – Menú principal interactivo .....	27
Figura 17 – Diagrama. OPCION 1) Crear Escenario .....	28
Figura 18 – Diagrama. OPCION 2) Destruir Escenario .....	29
Figura 19 – Diagrama. OPCION 3) Ejecutar Controlador .....	30
Figura 20 – Diagrama Monitor.py.....	32
Figura 21 – Monitor.py .....	33
Figura 22 – Menú de las posibles configuraciones .....	33
Figura 23 – Diagrama. OPCION 4) Realizar Configuración .....	34
Figura 24 – Menú interactivo para eliminar configuración IPv4.....	36
Figura 25 – Menú de las posibles visualizaciones .....	37
Figura 26 – Diagrama. OPCION 5) Visualizar Configuraciones .....	37

Figura 27 – Tabla de flujos del CONM_A.....	38
Figura 28 (a) Gráfica de Bandwidth (Kbps) en función del tiempo (segundos) del short_path	39
Figura 28 (b) Gráfica de Bandwidth (Kbps) en función del tiempo (segundos) del long_path	39
Figura 29 – Tabla de flujo del CONM_A por defecto .....	42
Figura 30 (a) Exp. 1. Ancho de banda utilizado del short_path en función del tiempo	42
Figura 30 (b) Exp. 1. Ancho de banda utilizado del long_path en función del tiempo	42
Figura 31 – Tabla de flujo del CONM_A luego de cambiar la configuración de ruteo .....	43
Figura 32 (a) Exp. 1. Ancho de banda utilizado del short_path en función del tiempo después de la configuración de caminos .....	43
Figura 32 (b) Exp. 1. Ancho de banda utilizado del long_path en función del tiempo después de la configuración de caminos .....	43
Figura 33 (a) Exp. 2. Ancho de banda utilizado del short_path en función del tiempo	44
Figura 33 (b) Exp. 2. Ancho de banda utilizado del long_path en función del tiempo	44
Figura 34 (a) Exp. 3. Ancho de banda utilizado del short_path en función del tiempo sin TE	45
Figura 34 (b) Exp. 3. Ancho de banda utilizado del long_path en función del tiempo sin TE	45
Figura 35 (a) Exp. 3. Ancho de banda utilizado del short_path en función del tiempo con TE	46
Figura 35 (b) Exp. 3. Ancho de banda utilizado del long_path en función del tiempo con TE	46
Figura 36 – Exp. 4. Pérdida de paquetes antes, durante y después de aplicar TE.....	47

## **Índice de tablas**

Tabla 1 – Direccionamiento L2 y L3 del escenario.....	22
Tabla 2 – Protocolos catalogados como tráfico sensible .....	40
Tabla 3 – Mapeo de puertos para utilizar en iperf.....	40
Tabla 4 – Resultados de iperf UDP cada 100 segundos .....	44
Tabla 5 – Comparativa de datos obtenidos con y sin TE ante pruebas de estrés.....	48

## **Índice de códigos y ecuaciones**

Código 1 – Extracto de configuración del contenedor VyOS “INTERNET” .....	20
Código 2 – Extracto de configuración del CONM_A.....	24
Ecuación 1 – Ecuación para la obtención del ancho de banda disponible .....	24
Código 3 – Extracto de código de script “Sensible_traffic_long_path.sh.....	26

## 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
MetroEthernet Forum	MEF
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
Underlay Conectivity Services	USC
Virtualized Network Service	VNS
Virtual Networks over Linux	VNX
Vector Packet Processor	VPP
Virtualized Services Directory	VSD
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 de mayor capacidad, soluciones novedosas de encaminamiento de paquetes o equipos mas potentes, son algunas de las soluciones que pueden permitirnos abordar ese problema. 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 han sufrido grandes cambios. Por otro lado, la red se vuelve cada vez más compleja, transportando un gran volumen de tráfico muy diverso. Esto dificulta la administración y gestión de la red, que es rígida, costosa y poco flexible casi por definición.

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 más 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 o Software-Defined Networks (SDN).

Consecuencia de este crecimiento es que se afianzan los ya conocidos Iass (Infrastructure as a Service), PaaS (Plataform as a Service) y SaaS (Software as a Service). Con esto, la migración de servicios alojados en la nube (tanto pública como privada) ha conducido a 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. Este TFM (Trabajo Fin de Máster) se centra en la aplicación de soluciones SDN a los entornos de redes WAN, es decir, a redes y servicios SD-WAN.

Tanto los proveedores de contenidos (Google, Microsoft, etc.) como los operadores de red ven la necesidad y el potencial de proporcionar servicios WAN definidos por software. De esta forma es que nacen 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 para garantizar los requisitos de las aplicaciones de negocio que se utilizan en una red corporativa que se compone de varias sedes remotas. En Internet, todos los usuarios compiten por el ancho de banda disponible y sus servicios se pueden ver afectados por el tráfico de otros usuarios. 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 trabajo 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”) que permita balancear los flujos de tráfico que atraviesan una red definida por software que une dos sedes remotas. Las sedes remotas cuentan con la particularidad de disponer con más una conexión WAN. Los clientes suelen demandar cierta calidad de servicio para priorizar el tráfico de sus principales aplicaciones de negocio y es aquí donde el proveedor SD-WAN debe ser capaz de configurar su red SDN para garantizar estos requisitos. Este TFM se centra en analizar cómo un operador puede optimizar esta conexión WAN ejecutando una aplicación SDN de ingeniería de tráfico. Sobre esta es que se hará el diseño de la TE.

La idea principal consiste, por un lado, en definir qué tráfico requiere mayores requisitos de calidad de servicio. Este es el que se desea priorizar respecto a los demás. Para ello se monitoreará la red del operador para determinar cómo se establece la conexión WAN internamente. Se programará una aplicación SDN que permita tomar decisiones en tiempo real de por dónde enviar el tráfico sensible sin afectar a la calidad de servicio de otros flujos de la red. Con esto se buscará obtener el mayor beneficio a las conexiones WAN de forma programática, escalable y rápida.

Una vez implementada la aplicación SDN se realizarán pruebas comparativas para determinar la utilidad de una solución de este tipo y entender finalmente el potencial que implica esto tanto para el usuario final como para la empresa proveedora de la conexión.

Para llevar a cabo este TFM, se diseñará un escenario virtual sobre VNX (Virtual Networks over Linux). VNX es una plataforma de código abierto implementada por el DIT (Departamento de Ingeniería de Telecomunicaciones) de la Universidad Politécnica de Madrid (UPM), que permite desplegar escenarios virtuales. En este escenario se definen los conmutadores SDN, que son gestionados en este caso por el controlador Ryu, el cual también es de código abierto y permite acoplarle una amplia gama de aplicaciones ya existentes. Una de estas es la llamada *Traffic Monitor*<sup>1</sup> la cual será de gran utilizad a la hora de obtener las prestaciones de los enlaces. Con esta aplicación SDN se puede extraer información en tiempo real como por ejemplo la cantidad de paquetes cursados por cada interfaz o el ancho de banda consumido en tiempo real.

---

<sup>1</sup> <https://github.com/muzixing/ryu>

Este TFM tiene dos contribuciones principales: la primera es el desarrollo de la aplicación SDN de ingeniería de tráfico que permite balancear la carga y priorizar el tráfico sensible sobre el resto. La segunda es la implementación de un controlador SD-WAN sencillo 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

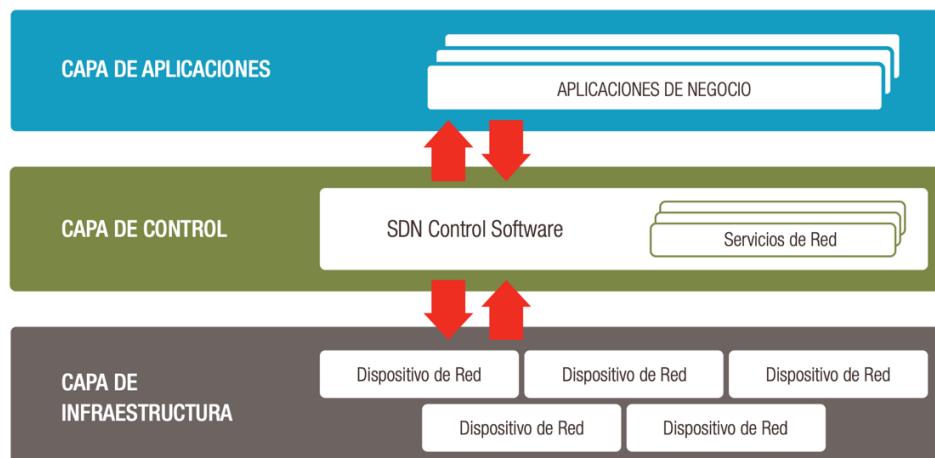
Las redes LAN (Local Area Network) son sin duda las mas utilizadas a nivel empresarial. Allí radican los servicios internos así como también los activos tales como impresoras, registros y servidores de información los cuales en un principio solo interesa que se tenga acceso desde dentro de la corporación. 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 se definió bajo el principio *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 asegurarse un nivel de QoS (Quality of Service) 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 enlace 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.

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 <sup>2</sup> del plano de datos <sup>3</sup> 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 1* [1] 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 con protocolos estándares como OpenFlow mediante paquetes *packet-in* <sup>4</sup> y *packet-out* <sup>5</sup> 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.



**Figura 1 - Arquitectura SDN [1]**

Con este nuevo paradigma un operador puede al cabo de unos pocos minutos levantar una nueva sede, realizar configuraciones remotas y establecer criterios para conmutar los flujos de tráfico, decidiendo en tiempo real el camino mas óptimo por el

<sup>2</sup> Orientado a tratar el tráfico orientado a la gestión, mantenimiento y modificación de la red y sus componentes.

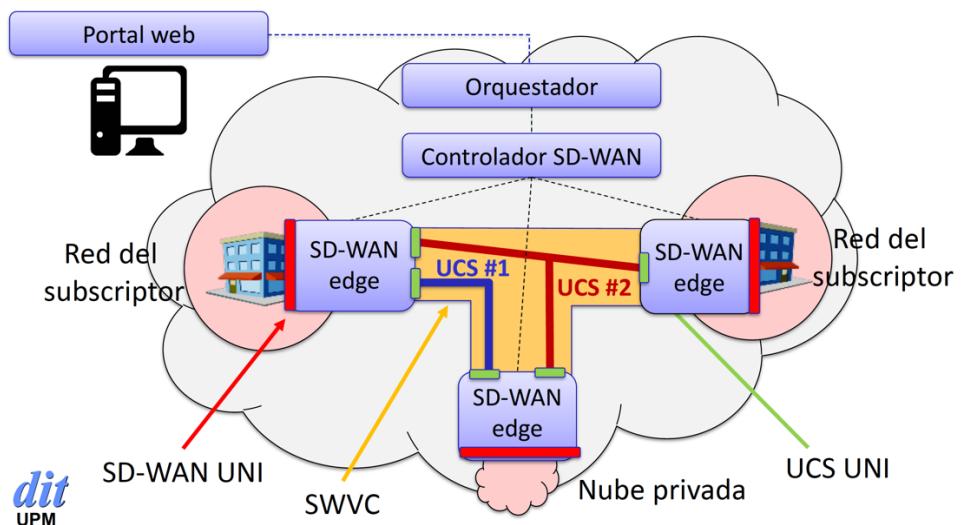
<sup>3</sup> Orientado a tratar el tráfico destinado a los servicios.

<sup>4</sup> Paquetes del conmutador al controlador.

<sup>5</sup> Paquetes del controlador al conmutador.

cual enviar los paquetes en función de los requisitos. A su vez, con la inclusión de la virtualización de funciones de red (Network Functions Virtualization o NFV) es posible realizar un despliegue de equipamiento de red virtualizado como si fuera físico. Esto disminuye el tiempo de despliegue 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 operador de la red es o no el mismo que brinda la solución SD-WAN. En definitiva, permite unificar las conexiones WAN y tratarlas como si fuera “una sola” bajo demanda, realizando balanceos de carga entre ambas conexiones de ser necesario.

Como se mencionó anteriormente, es común que las empresas suelan tener mas de una conexión de red hacia el exterior. Mediante UCS (Underlay Connectivity Services) se interconectan las redes de los suscriptores permitiendo así el acceso a sus aplicaciones de negocio, a sitios remotos o directamente a internet tal cual se muestra en la *Figura 2*. El concepto principal de una arquitectura MEF (MetroEthernet Forum) radica en la existencia de routers SD-WAN Edge ubicados en los sitios del suscriptor los cuales cuentan con conexiones lógicas hacia el controlador SD-WAN y físicas hacia una o varias redes WAN según corresponda.



*Figura 2 – Arquitectura MEF [2]*

Estas conexiones WAN suelen ser dedicadas en función de los objetivos para los cuales están diseñadas ya que en muchos casos se requiere de una seguridad y calidad de servicio extra que Internet no ofrece. Para estos casos es necesaria una implementación en la cual permita coexistir redes privadas (MPLS por los cuales se levantan túneles privados) y públicas como es el caso de Internet.

En un escenario como se plantea en la *Figura 2* pueden convivir y trabajar en conjunto mas de un proveedor de red a la vez. En este caso se distinguen claramente dos (uno por

cada UCS) y además quien realiza la gestión SD-WAN con los SD-WAN Edges, controlador y orquestador podría llegar a ser otro proveedor distinto. En este caso el administrador de la solución SDN tiene la capacidad, mediante el controlador, de decidir por cuál túnel de la red enviar los paquetes. Por otra parte, cuenta con la capacidad de automatizar el despliegue, gestión, monitorización y operación de la red mediante aplicaciones ad-hoc.

En caso de que sea el mismo proveedor quien proporcione la gestión SD-WAN y los UCS's, éste tiene además la posibilidad de poder establecer los caminos de los paquetes por los túneles de forma de brindar mayores prestaciones al cliente en función de sus requisitos. Este es precisamente el caso de estudio e implementación de este TFM. Aquí se supondrá que el administrador de la solución SD-WAN es el mismo que opera la red WAN, de forma que será posible no solo decidir el túnel por le cual enviar el tráfico, sino que también aplicar una solución avanzada de ingeniería de tráfico para en encaminar los paquetes en tiempo real y de la manera más óptima.

## 2.1 Soluciones comerciales académicas

La flexibilidad que proporcionan las redes SDN en los entornos WAN ha capturado la atención de proveedores de contenidos y operadores de red. Actualmente, existen diversas soluciones SD-WAN pero en la mayor parte de los casos, son servicios de pago. Esto dificulta el análisis de estas soluciones en los entornos académicos. 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, a día de hoy, es una solución de código abierto. En este TFM se han realizado pruebas sobre FlexiWAN, las cuales se detallan en el capítulo 4.1 de este documento.

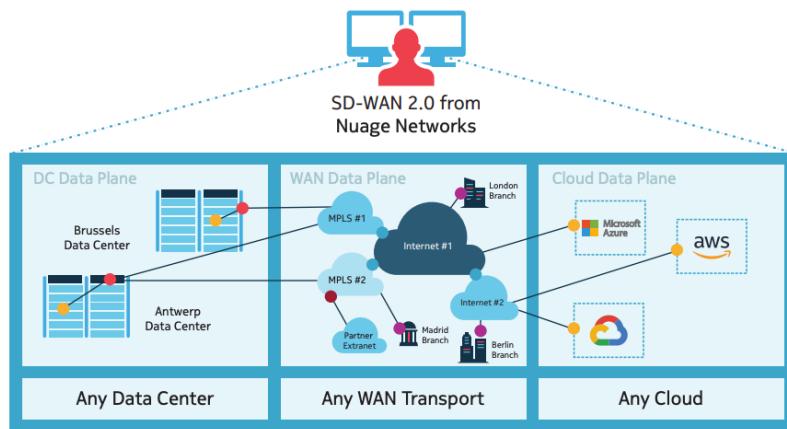
### 2.1.1 Nuage

**Nuage Networks** [3] es la solución SD-WAN implementada por Nokia. Permite automatizar la conexión entre diferentes sedes remotas y gestionar las conexiones WAN del cliente mediante la definición de políticas en los nodos que se instalan en las sedes del suscriptor, los nodos SD-WAN Edge. Esto disminuye los costes del suscriptor que puede aprovechar la conexión WAN pública de una forma más eficiente. También disminuye los costes del proveedor del servicio WAN, puesto que facilita la gestión y configuración de los equipos de la red SD-WAN.

Con VNS (*Virtualized Network Services*) de Nuage, la red SD-WAN se puede optimizar dinámicamente para encaminar el tráfico por la conexión WAN mas adecuada. En muchos casos las empresas suelen tener conexiones IP, MPLS, 3G y/o LTE

de diferentes proveedores para transportar el tráfico de su corporación. Los servicios SD-WAN permiten utilizar la conexión a Internet para encaminar el tráfico de aplicaciones que no requieran requisitos de calidad de servicio exigentes. Así, solo se utiliza la conexión WAN privada para servicios críticos. El hecho de que estos enlaces sean provistos por diferentes ISP es indiferente 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, tal y como muestra la *Figura 3*.



**FIGURE 2. Creating a seamless underlay network with SD-WAN 2.0**

(● NSG - Underlay border router   ● NSG - Virtualized   ● NSG - Border router   ● NSG - uCPE   ● NSG - Underlay border router)

**Figura 3 – Red sin interrupciones SD-WAN [3]**

La *Figura 4* muestra la arquitectura de Nuage. El VSD (Virtualized Services Directory) da al administrador SD-WAN la posibilidad de definir y aplicar políticas a la red de una manera fácil, gestionada y centralizada. Además, permite recolectar datos de la red referidos a paquetes cursados, solicitudes a aplicaciones o tasa de tráfico de forma periódica para la creación de informes, así como también alertas en caso de que el 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 entre VNF's de firewalls, IPSec, NAT, balanceo de carga y gestión de dominios, entre otros.

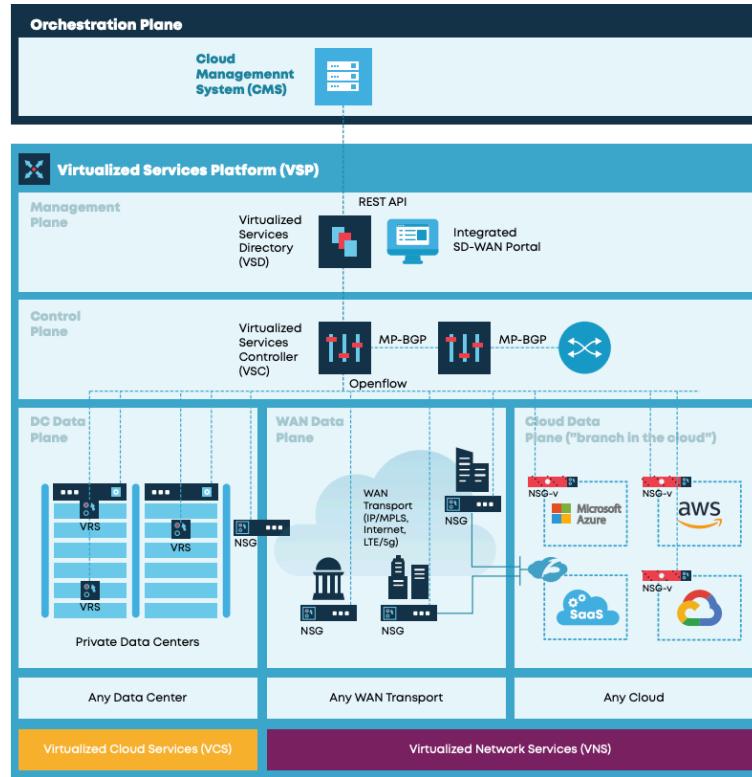


Figura 4 – Arquitectura Nuage [3]

### 2.1.2 Viptela

**Viptela** [4] es la solución SD-WAN del fabricante Cisco. Viptela proporciona una visión en tiempo real del estado de la red que utiliza para tomar decisiones de encaminamiento. Está conformada por cuatro componentes: vEdge, vSmart, vManage y vBond los cuales se ven en la *Figura 5*.

El nodo *vEdge* representa el router que se despliega en las sedes remotas de la red corporativa. Se trata del nodo SD-WAN Edge el cual se encarga de establecer comunicaciones seguras con los demás vEdges (virtualizados o físicos) mediante túneles IPSec y ejecutan las políticas que implementa el controlador *vSmart*.

El nodo *vSmart* es el controlador de la red SD-WAN. Se encarga de establecer conexiones SSL con los demás componentes que forman la arquitectura SD-WAN mediante un protocolo propietario llamado OMP (Overlay Management Protocol). Este proporciona seguridad, control de acceso y funciones de encaminamiento sin la utilización de protocolos tradicionales como OSPF y BGP. En cuanto a las políticas, éstas 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*. Permite a los operadores de red realizar configuraciones, resolver problemas, planificar actividades de aprovisionamiento y monitorear la red SD-WAN.

vManage soporta diversos protocolos de gestión como SNMP, NETCONF o Syslog. Además, cuenta con la capacidad de visualizar por medio de la interfaz las estadísticas e información recopilada del DPI (Deep Packet Inspection) hechas en los vEdges. Permite reconocer y clasificar el tráfico en más de 1000 aplicaciones diferentes y plasmarlas en el panel central por categorías y sub-categorías para que el operario tenga una visión rápida y clara del tráfico que se cursa por su red.

El último elemento de Viptela es el *vBond* que cumple con el rol de orquestador SD-WAN. vBond se encarga de realizar la autenticación y autorización de los elementos de la red gestionando las comunicaciones entre los vEdges y el controlador.

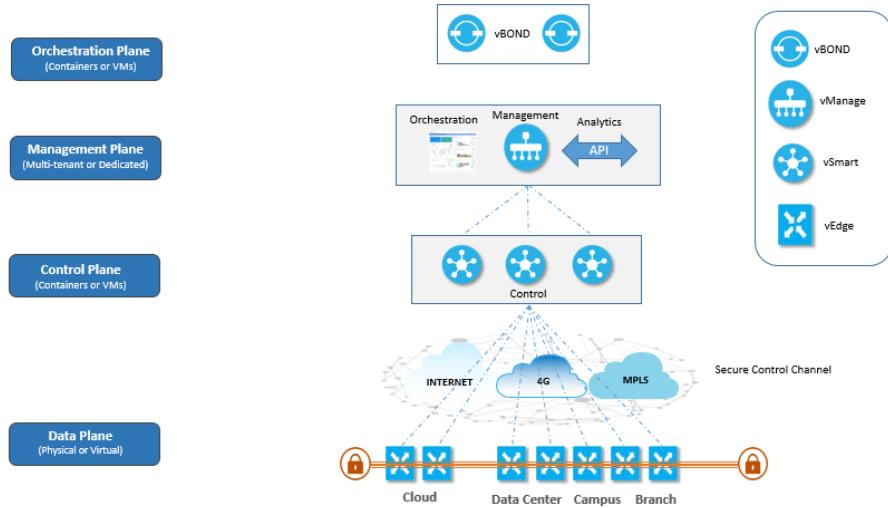


Figura 5 – Arquitectura Viptela [4]

### 2.1.3 FlexiWAN

**FlexiWAN** [5] es otra solución SD-WAN que a diferencia de las anteriores, es de código abierto. FlexiWAN permite instalar hasta 3 SD-WAN Edge de forma gratuita aunque con ciertas limitaciones. La arquitectura de FlexiWAN está formada principalmente por dos actores principales, tal y como se ve en la Figura 6: los nodos *FlexiEdge* y *FlexiManage*. El FlexiEdge es el dispositivo que se despliega en las dependencias de los suscriptores. Uno de los componentes software que tiene instalado el FlexiEdge es VPP (Vector Packet Processor). Este es un código de producción que se encuentra ejecutado en los FlexiEdge, el cual permite aumentar los límites de rendimiento y velocidad de procesamiento de los paquetes ya que puede procesar varios paquetes en paralelo. Por otro lado, FlexiEdge tiene una suite llamada FRR (Free Range Routing) que implementa diferentes protocolos de encaminamiento como OSPF, RIP, BGP o IS-IS. Los nodos FlexiEdge se conectan a FlexiManage, quien se encuentra ejecutándose en la nube del proveedor del servicio. A través de FlexiManage, un administrador puede gestionar y recopilar información de los dispositivos FlexiEdge para luego analizarla y realizar 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 actuar en el menor tiempo posible.

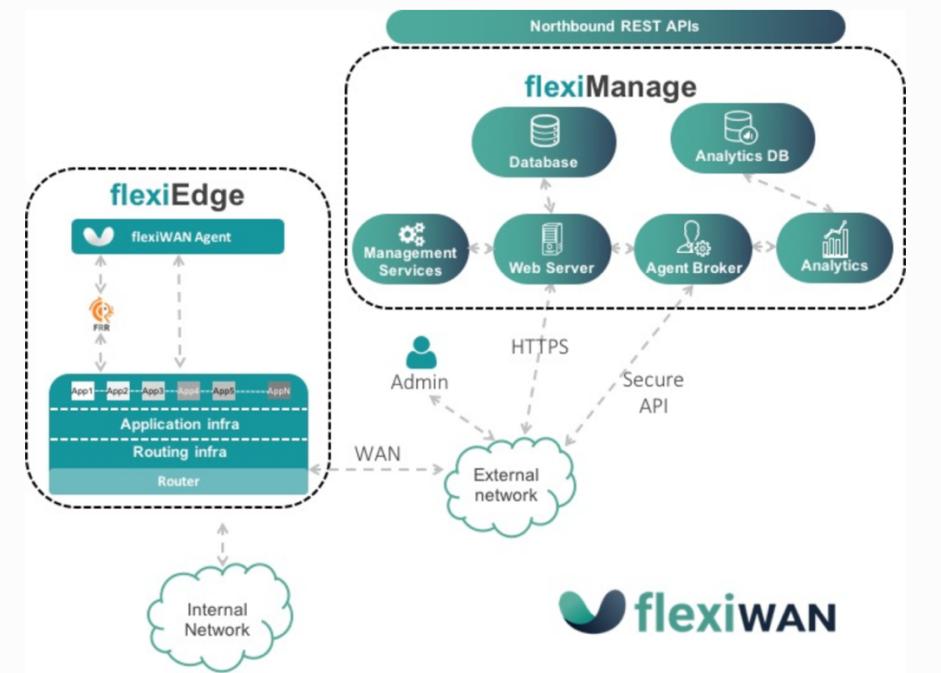


Figura 6 – Diagrama de bloques de alto nivel de la arquitectura FlexiWAN [5]

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

Las soluciones comerciales descritas en el apartado anterior muestran la diversidad que existe a la hora de diseñar soluciones SD-WAN para tener valor añadido con respecto a sus competidores. 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. Este TFM está inspirado en la solución que propone Google para conectar sus centros de datos a través de una conexión WAN basadas en conmutadores SDN.

A rasgos generales, la ingeniería de tráfico consiste en encaminar el tráfico a partir de las condiciones 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 multicamino.
- Reducir el retardo extremo a extremo: Algoritmos de encaminamiento como CSPF (Constrained Shortest Path First) permiten determinar el camino mas corto entre dos puntos de la red, sujeto a unos determinados requisitos de retardo. Este protocolo de estado de enlace es utilizado cuando se requiere brindar calidad de

servicio ya que el camino óptimo lo obtiene a partir de métricas como ancho de banda, delay o cantidad de saltos.

- Minimizar la pérdida de paquetes: Se puede dar por diversos motivos, como por ejemplo, por la caída o saturación de los enlaces y nodos de conmutación. Para disminuir este factor de riesgo se suelen emplear caminos de respaldo para disminuir al máximo el problema.
- Priorización de tráfico: Es posible aplicar técnicas para determinar la prioridad del tráfico, con el objetivo de tratarlo de forma diferenciada. Esto permite conmutar con mayor prioridad los paquetes que son más sensibles a requisitos de QoS.

Si bien mediante el encaminamiento en redes MPLS es posible cumplir con algunas de estos puntos, sigue habiendo algunas limitaciones para aplicar estas técnicas. Ejemplo de esto es el inconveniente que trae asociado MPLS-TE a la hora del congestionamiento en escenarios multicamino. MPLS-TE requiere de una reordenación excesiva de paquetes. El desorden puede llegar a acabar en un problema de congestion avoidance, produciendo degradación de throughput de las aplicaciones.

Existen diversas soluciones de ingeniería de tráfico que se han diseñado para ser aplicadas en conexiones WAN basadas en redes de conmutación SDN. Los ejemplos más representativos de estas soluciones son B4 y SWAN propuestas por Google y Microsoft respectivamente. SWAN define clases de tráfico y a cada una de ellas le asigna un ancho de banda determinado en función de los requisitos de la aplicación. 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. Este TFM comparte las ideas de B4 en el sentido de que propone una solución de ingeniería de tráfico sobre una red WAN basada en SDN.

### 2.2.1 B4

**B4** [6] es la solución de SD-WAN implementada por Google. Permite interconectar dos sedes remotas mediante una red SDN, es decir, a través de conmutadores OpenFlow. La idea primaria de B4 consiste en aplicar soluciones de ingeniería de basadas en algoritmos de encaminamiento multicamino que tienen en cuenta los requisitos de QoS de las aplicaciones. Por otro lado, B4 resuelve el problema de adaptar la asignación de ancho de banda a la demanda variable de las aplicaciones y a posibles caídas en los enlaces y conmutadores.

El algoritmo permite determinar la ruta que deben seguir los datos de la aplicación en la red para que se satisfagan sus requisitos de QoS. Esta asignación se realiza en función de las prioridades de las aplicaciones, las cuales vienen definidas a partir de un ancho de banda demandado y la sensibilidad frente a retardos.

### 3 Casos de uso

En este capítulo se van a explicar dos posibles casos de uso donde las soluciones del TFM tienen cabida. El caso de uso A consiste en una solución conformada por 3 operadores de red y donde la ingeniería de tráfico se debe aplicar en los SD-WAN Edge. Esto se debe a que el proveedor del servicio SD-WAN no es quien provee el servicio de conexión WAN. El caso de uso B tiene la particularidad de aquí el operario de la red WAN es el mismo del que ofrece el servicio SD-WAN, con lo cual la ingeniería de tráfico se realiza en la red de commutadores.

#### 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 X con la sucursal mas pequeña que se encuentra en la ciudad Y. 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 para que los trabajadores de la ciudad Y tengan la misma calidad de experiencia y servicio que tienen los trabajadores de la ciudad X. A raíz de ello, la empresa en cuestión tiene contratados dos conexiones WAN. Una, a cuyo proveedor lo llamaremos *proveedor A*, que brinda conectividad a Internet el *proveedor B* proporciona, la conexión WAN privada y el *proveedor C* el servicio SD-WAN.

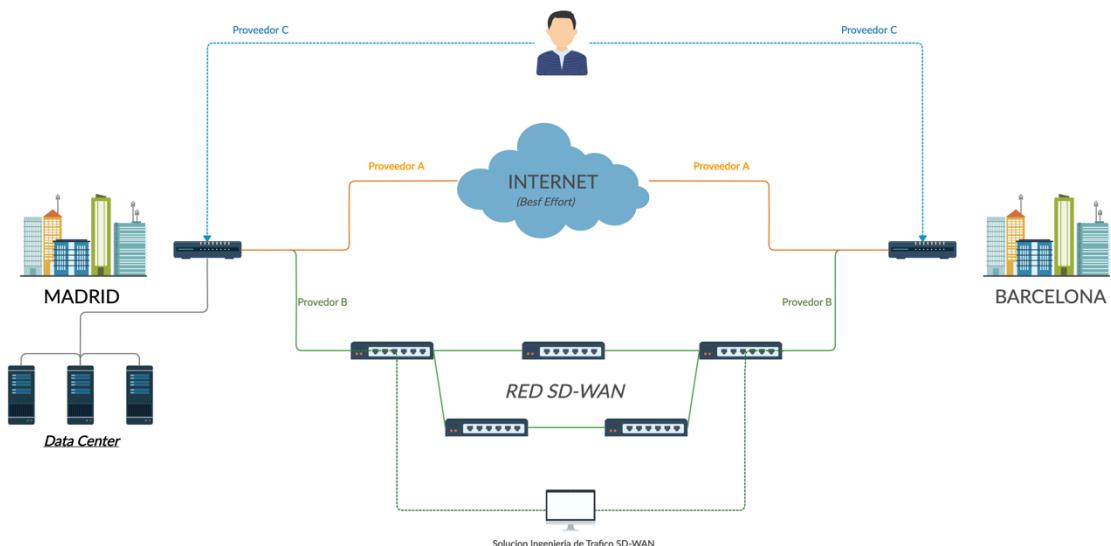
El proveedor del servicio SD-WAN y el operador de la conexión WAN pueden ser distintos actores, tal y como refleja el estándar del MEF de la *Figura 2*. Para el caso de la *Figura 7* el proveedor C proporciona únicamente servicios SD-WAN pero no tiene potestad para la configuración de las rutas a nivel de los commutadores del proveedor B. De esta forma, el proveedor C se encargaría únicamente de gestionar y encaminar el tráfico por las UCS's entre las sedes remotas mediante configuración de los SD-WAN Edge. En un escenario de este tipo la ingeniería de tráfico se aplica en el borde pero no sobre la red WAN en sí misma, ya que al ser de otro proveedor, este no tiene permisos de hacerlo.

Para este caso supondremos que el proveedor B y C es el mismo, con lo cual este podrá a parte de elegir el enlace WAN a utilizar, también podrá definir los caminos dentro de su red privada para este cliente. El suscriptor tiene unos requisitos de QoS que indica al proveedor del servicio SD-WAN y este configura los nodos SD-WAN Edge y la conexión WAN privada para que se cumplan dichos requisitos. Para configurar la conexión WAN se deben aplicar soluciones de ingeniería de tráfico (TE) en su red, con el objetivo de encaminar el tráfico de las aplicaciones más sensibles a los requisitos de QoS. 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 medio de soluciones SD-WAN, el operador de la red tiene la posibilidad de optimizar la red WAN, en busca de satisfacer las necesidades requeridas por el cliente. Mediante implementaciones TE, el administrador puede configurar los túneles para realizar un mejor tratamiento de los paquetes en función de las necesidades solicitadas y el contrato firmado con el cliente.

La red del proveedor B suele ser compleja ya que busca interconectar varios sitios remotos de diversas empresas. En este TFM se ha estudiado un problema conocido en la literatura como “el problema del pez”, que se basa en una topología de red en la que OSPF presenta algunas limitaciones.

Hay que tener en cuenta que este trabajo se centra en la optimización de la conexión WAN privada, por lo que no se estudia cómo se pueden utilizar múltiples enlaces WAN de forma simultánea.



**Figura 7 - Escenario modelo de caso de uso A**

### 3.2 Caso de uso B

El segundo caso de uso donde se puede aplicar este trabajo es a la hora de interconectar dos o más centros de datos en una red SD-WAN multicamino. Debido al tipo de tráfico que se intercambia, se requieren enlaces que soporten un gran ancho de banda y transferencias de información bajo demanda y con comportamientos elásticos. Sin embargo diseñar enlaces dedicados exclusivamente para esto son excesivamente costosos. Es por ello que es necesario aplicar soluciones de ingeniería de tráfico que permitan optimizar el uso de los canales, al mismo tiempo que se brinda calidad de servicio para las aplicaciones que lo requieran.

Se supone un escenario como se detalla en la *Figura 8* 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 constantes en el tiempo ni igualmente prioritarios, sino que haya servicios que requieran de mayores prestaciones que otros. Para este caso es necesario aplicar TE de forma de priorizar algunos tipos de datos, ya sea teniendo en cuenta las direcciones MAC, IP o puertos de origen y/o destino. Otra forma de categorizar el tráfico es por clases como lo hace B4, donde se tiene en cuenta el volumen y la sensibilidad ante retardos. 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 una ingeniería de tráfico.

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. De no ser así se haría poco escalable.

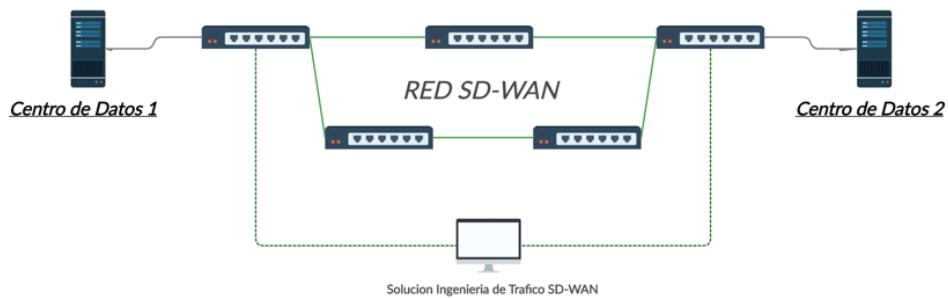


Figura 8 – Escenario modelo de caso de uso 2

## 4 Aplicación de ingeniería de tráfico SD-WAN

En esta sección se procede a detallar en profundidad la implementación de una solución SDN de ingeniería de tráfico que se puede utilizar en los casos de uso definidos en el capítulo 3. Por otro lado se mostrará el desarrollo de un controlador SD-WAN con una interfaz sencilla mediante la cual el administrador sea capaz de realizar configuraciones y visualizaciones de la red en tiempo real. A lo largo del capítulo se irán comentando y justificando las decisiones tomadas, las herramientas utilizadas y la teoría aplicada para el diseño de esta ingeniería de tráfico.

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 un tráfico determinado a partir de dos componentes esenciales: el ancho de banda libre de los canales y el tipo de tráfico a enviar.

Para obtener los parámetros de la red fue necesario utilizar una aplicación del controlador llamada *Traffic Monitor*, la cual fue levemente modificada para obtener numéricamente el valor de ancho de banda libre a partir del valor de la capacidad del canal y el uso instantáneo por el mismo. Estas modificaciones y mas información sobre el *Traffic Monitor* se detallan en la sección 4.3.1 de este documento.

El caso más habitual es que el controlador WAN solo sepa lo que pase en los SD-WAN Edge pero no así en la red WAN misma. De esta forma tiene potestad mediante ingeniería de tráfico en el borde de la red WAN, elegir el UCS por el cual enviar los paquetes. Sin embargo, este caso de estudio es un poco diferente ya que de acuerdo a como está planteada esta solución TE, el controlador SD-WAN si posee información de la red WAN. Aquí la solución de ingeniería no está aplicada sobre los SD-WAN Edge sino que lo está sobre la red misma de comutadores.

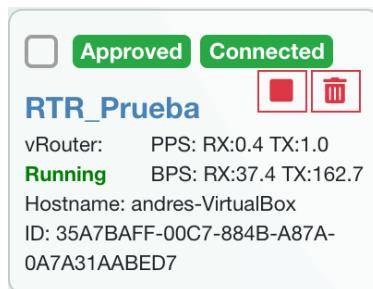
El escenario a tratar (sección 4.2) busca parecerse lo mas posible al caso de uso de la sección 3.1 donde para llegar de un origen a un destino puede haber mas de un camino posible y es aquí donde el controlador debe decidir por cual de ellos debe enviar los paquetes. Se entiende por “tráfico sensible” todo aquel que requiera de ciertos requisitos para su funcionamiento y haya sido previamente definido el administrador de red. En el apartado 4.5 de este documento se detallan ciertos tipos de tráficos sensibles a tener en cuenta.

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

Una vez estudiada la teoría de FlexiWAN [5] en el punto 2.1.3 de este documento, se procedió a implementar un escenario para trabajar con la solución comercial FlexiWAN. En primer lugar, se crearon y configuraron en VirtualBox dos máquinas virtuales Ubuntu y sobre ellas se instalaron los routers virtuales (FlexiEdge). Estas máquinas contaban con una interfaz con conexión a Internet cada una mediante la cual

se conectaban al FlexiManage. El FlexiManage se encuentra corriendo en la nube y es quien se encarga como se mencionará mas adelante, de centralizar las configuraciones y visualizaciones de la red.

Para el que FlexiManage pueda reconocer a los FlexiEdge como sitios seguros y mostrarlos en su dashboard, fue necesario asociarles a estos un token (creado desde la interfaz del 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 9*. Una vez que su estado se encuentra en “Running” quiere decir que se configuró correctamente.



**Figura 9 – Visualización el FlexiEdge desde el FlexiManage**

Una vez configuradas las sedes remotas, en FlexiManage se pueden establecer diferentes configuraciones. Por ejemplo cambiar las direcciones IP públicas y privadas del FlexiEdge o crear nuevas interfaces. Todo esto se hace de forma centralizada y en pocos minutos lo cual proporciona una flexibilidad y rapidez que las redes WAN tradicionales no pueden igualar. Por otro lado permite la creación de túneles virtuales entre los FlexiEdges para proporcionar conectividad entre las sedes remotas. Una vez hecho esto la herramienta permite medir parámetros como la latencia y la tasa de paquetes perdidos por estos túneles.

Desde FlexiManage también es posible obtener las tablas de encaminamiento de cada FlexiEdge y modificar o agregar nuevas rutas en caso de ser necesario. También permite visualizar estadísticas en tiempo real del tráfico cursado, tanto en BPS (Bits Por Segundo) o PPS (Paquetes Por Segundo). Esto se muestra en la *Figura 10* a modo de ejemplo.

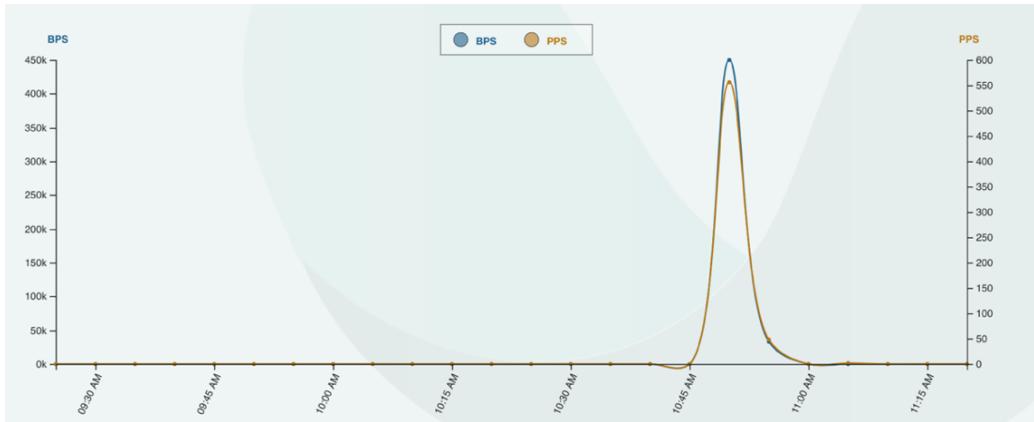


Figura 10 – Estadísticas de bps y pps en un FlexiWAN-router

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 Escenario virtual

El escenario planteado para la realización de este trabajo fue elaborado mediante la herramienta de virtualización de redes VNX. En particular, se ha definido el escenario utilizando el lenguaje de especificación que utiliza VNX, es decir, se ha creado un fichero XML que define el escenario. Para su diseño se tuvo en cuenta que la red WAN privada debía poder admitir múltiples caminos.

Se realizó la configuración de las interfaces tanto en IPv4 como IPv6. De esta forma se tiene conectividad mediante ambos protocolos de red y todas las configuraciones a nivel de controlador se realizaron para ambos como se puede apreciar mas adelante en los diagramas de flujo de la sección 4.4 donde se especifican los scripts utilizados.

La topología de la red de commutación (WAN privada) se representa en la *Figura 11* y se conoce como topología en forma de “pez”. Está compuesta por los commutadores OVS A,B,C,D y E y routers containers de Linux llamados R1\_edge y R2\_edge que simulan ser SD-WAN Edges. Aquí se puede ver que para encaminar un paquete desde R1\_edge a R2\_edge por la red de commutadores 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 múltiples caminos sino en que uno de ellos es mas largo que el otro.

Por otra parte, se模拟aron dos sedes remotas, cada una con dos hosts los cuales se encuentran configurados sobre contenedores LXC. Estos sistemas finales llevan los nombres de “h01”, “h02”, “h03” y “h04” estando los primeros dos en un sitio remoto y los restantes en el otro.

Siguiendo con el escenario, se ha definido un contenedor LXC-VyOS de Linux que simula la conexión WAN pública. Este contenedor se puede ver en el escenario bajo el nombre de “INTERNET”. 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 camino que involucra a “INTERNET” ya que los SD-WAN Edge están configurados para que encaminen todos los paquetes hacia la red de commutadores (conexión WAN privada). Esto se debe a que el objetivo del TFM consiste en realizar una ingeniería de tráfico sobre una red de commutadores y no sobre los SD-WAN Edges, con lo cual todo el tráfico en este caso es necesario que vaya por esta red.

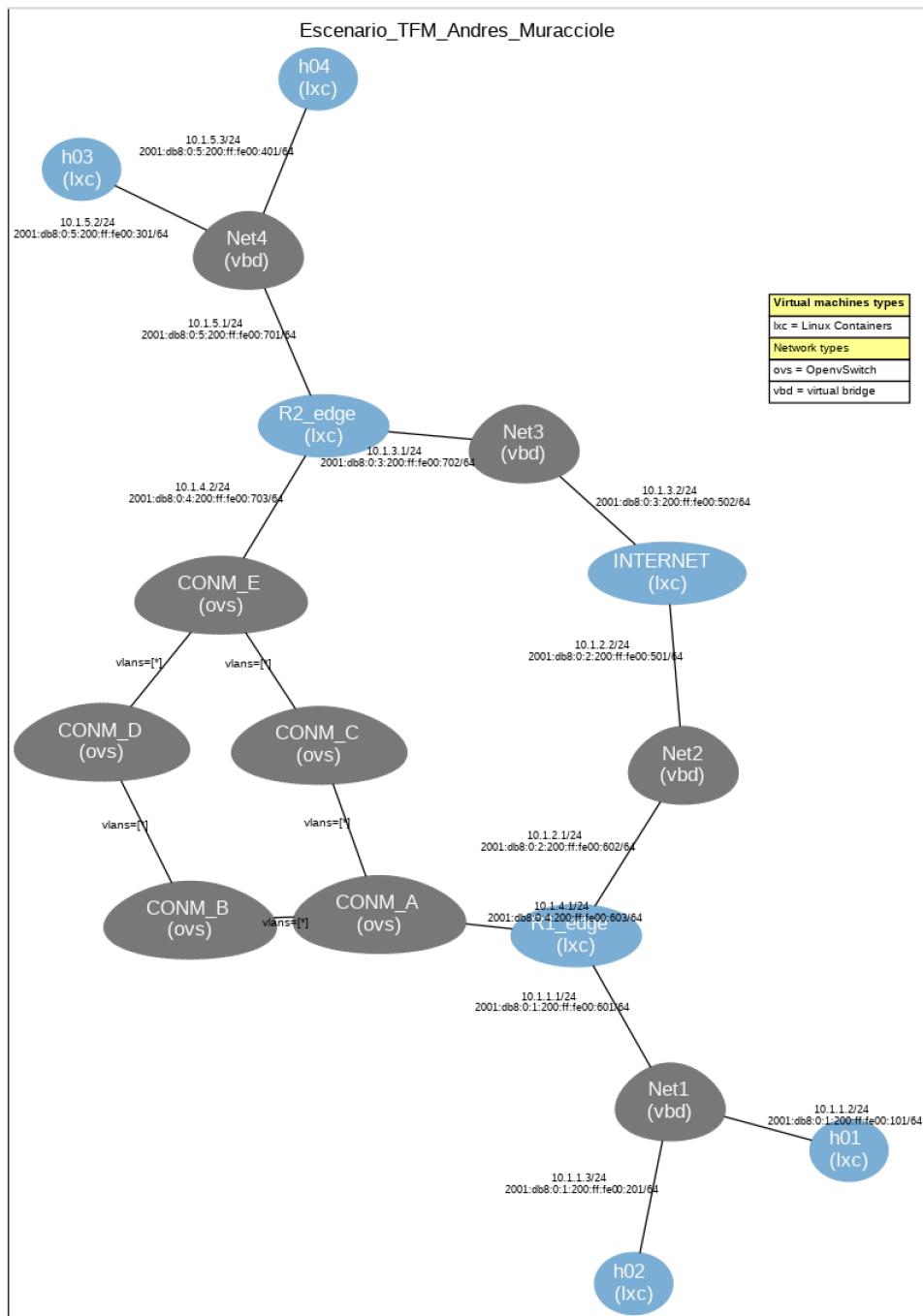


Figura 11 – Escenario virtual con red de commutadores en topología de pez

Como se mencionó anteriormente, si bien en camino por la red llamada INTERNET no se usa ya que todo es dirigido a la red de conmutadores, esta igualmente está configurada para dejar pasar el tráfico entre los SD-WAN Edge. Esto se hace mediante la línea 14 del código que se muestra a continuación. Aquí se refleja la configuración del contenedor INTERNET en el archivo xml del escenario. Con ella todo el tráfico que llega por una interfaz, lo manda por la otra haciendo simplemente un forwarding.

```

1 <vm name="INTERNET" type="lxc" arch="x86_64">
2   <filesystem type="cow">/usr/share/vnx/filesystems/rootfs_lxc_ubuntu64</filesystem>
3   <if id="1" net="Net2" >
4     <mac>00:00:00:00:05:01</mac>
5     <ipv4>10.1.2.2/24</ipv4>
6     <ipv6>2001:db8:0:2:200:ff:fe00:501/64</ipv6>
7   </if>
8   <if id="2" net="Net3" >
9     <mac>00:00:00:00:05:02</mac>
10    <ipv4>10.1.3.2/24</ipv4>
11    <ipv6>2001:db8:0:3:200:ff:fe00:502/64</ipv6>
12  </if>
13  <exec seq="on_boot" type="verbatim">
14    sudo sysctl net.ipv4.ip_forward=1
15  </exec>
16 </vm>
```

#### Código 1 - Extracto de configuración del contenedor VyOS “INTERNET”

VyOS es un sistema operativo de código abierto de red basado en Debian GNU/Ubuntu que permite ser ejecutado en hardware, máquinas virtuales o contenedores. En este caso se implementó en un contenedor para que oficie tanto e SD-WAN Edge como de red INTERNET según corresponda.

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 12*. Esto se debe a que tiene menor métrica, dejando inutilizable el camino mas largo. De esta forma, a pesar de tener un camino por el cual no se está traficando datos, se puede llegar a saturar la red ya que solo se utiliza un camino [7].

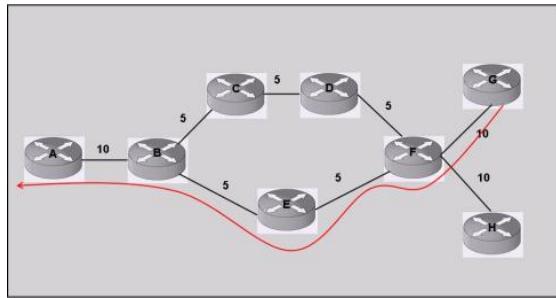


Figura 12 - Topología de pez

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 13*. Sin embargo esto no 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.

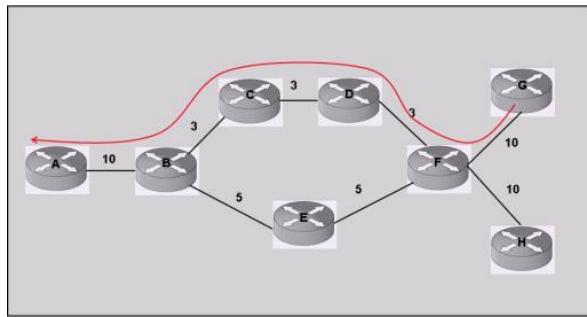


Figura 13 - Topología de pez

La manera de solucionar esto y utilizar ambos caminos es definiendo manualmente los costes de los enlaces de tal forma que para ambos caminos la suma sea igual y de esta forma utilizar los dos. Sin embargo, en esta trabajo no se busca resolver el balanceo de carga de dicha forma sino que se hará dinámicamente mediante entradas en las tablas de flujo de los conmutadores a partir de los requisitos de calidad de servicio necesarios. 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, volviendo una solución mucho mas granular y específica.

Los componentes que conforman el escenario de la *Figura 11* se encuentran configurados como se muestra en la *Tabla 1*:

NOMBRE	INTERFAZ	MAC	IPv4	IPv6
h01	1	00:00:00:00:01:01	10.1.1.2	2001:db8:0:1:200:ff:fe00:101
h02	1	00:00:00:00:02:01	10.1.1.3	2001:db8:0:1:200:ff:fe00:201
h03	1	00:00:00:00:03:01	10.1.5.2	2001:db8:0:5:200:ff:fe00:301
h04	1	00:00:00:00:04:01	10.1.5.3	2001:db8:0:5:200:ff:fe00:401
R1_edge	1	00:00:00:00:06:01	10.1.1.1	2001:db8:0:1:200:ff:fe00:601

R1_edge	2	00:00:00:00:06:02	10.1.2.1	2001:db8:0:2:200:ff:fe00:602
R1_edge	3	00:00:00:00:06:03	10.1.4.1	2001:db8:0:4:200:ff:fe00:603
R2_edge	1	00:00:00:00:07:01	10.1.5.1	2001:db8:0:5:200:ff:fe00:701
R2_edge	2	00:00:00:00:07:02	10.1.3.1	2001:db8:0:3:200:ff:fe00:702
R2_edge	3	00:00:00:00:07:03	10.1.4.2	2001:db8:0:4:200:ff:fe00:703
INTERNET	1	00:00:00:00:05:01	10.1.2.2	2001:db8:0:5:200:ff:fe00:501
INTERNET	2	00:00:00:00:05:02	10.1.3.2	2001:db8:0:3:200:ff:fe00:502
CONN_A	-	00:00:00:00:0A:00 (VNX hwaddr)	-	
CONN_B	-	00:00:00:00:0B:00 (VNX hwaddr)	-	
CONN_C	-	00:00:00:00:0C:00 (VNX hwaddr)	-	
CONN_D	-	00:00:00:00:0D:00 (VNX hwaddr)	-	
CONN_E	-	00:00:00:00:0E:00 (VNX hwaddr)	-	

Tabla 1 - Direccionamiento L2 y L3 del escenario

Usuario y contraseña de los host: root/xxxx

Usuario y contraseña de R1\_edge, R2\_edge y contenedor INTERNET: vyos/vyos

Los commutadores se conectan al controlador Ryu mediante el protocolo OpenFlow 1.3. Todo lo relacionado al controlador se procede a explicar con mayor detalle en el capítulo 4.3.

#### 4.2.1 Wondershaper

Una forma sencilla de poder limitar en Linux el ancho de banda de las interfaces (sean físicas como virtuales) es mediante el programa Wondershaper<sup>6</sup>. Este permite asignar de forma individual un límite de subida y bajada expresado en Kbps en las interfaces existentes. Para este trabajo se creó un script el cual se ejecuta de forma automática al levantar el escenario y especifica un ancho de banda máximo de 100 Mbps en las interfaces “A-C-0” y “A-B-1” tanto de subida como de bajada. Con esto se asegura que no habrá mas de este ancho de banda en la red SDN. Este valor fue elegido como prueba de concepto para poder realizar las simulaciones del capítulo 5 pero se podría haber fijado otro valor simplemente modificando el script “Wondershaper.sh”.

### 4.3 Controlador Ryu

El controlador elegido para esta implementación es Ryu<sup>7</sup>. En primer lugar, porque es una solución Open Source que cuenta con una gran cantidad de aplicaciones creadas por desarrolladores y publicadas en diversos portales abiertos y foros reconocidos. Esto potencia el uso de la plataforma ya que se genera un clima acorde

<sup>6</sup> <https://www.tecmint.com/wondershaper-limit-network-bandwidth-in-linux/>

<sup>7</sup> <https://ryu-sdn.org/index.html>

para la transferencia de conocimientos sobre el controlador. En segundo lugar, porque soporta NETCONF, extensiones Nicira y diversas versiones de OF (OpenFlow) [8], en especial la 1.3 que es la utilizada en este TFM. Ryu proporciona una API REST básica pero a la vez potente para diseñar soluciones SDN acordes a cada desarrollador.

Utilizando Ryu es posible configurar la tabla de flujos de los conmutadores de forma proactiva o reactiva. La programación proactiva consiste en configurar las tablas antes de que los paquetes de datos lleguen 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 procesamiento. Esto se hace mediante la API REST propia de Ryu. Por otro lado, la programación reactiva se produce cuando el conmutador recibe un paquete que no coincide con ninguna de sus entradas en su tabla de flujo y debe realizar una consulta al controlador mediante un mensaje de control de tipo *packet-in*. El controlador responde al conmutador con un *flow-mode* para que este último modifique el estado de sus tablas. Esta modificación puede ser por ejemplo agregar, modificar o quitar una entrada de su tabla de flujo. La desventaja del procedimiento reactivo frente al proactivo es que a gran escala genera un tráfico considerable en la red así como también un mayor retardo (tiempo de paquetes del controlador al conmutador y viceversa, tiempos de procesamiento de los paquetes y tiempo de llenado de tablas de flujo) [9].

El código del controlador Ryu se encuentra programado en Python y está alojado en Github<sup>8</sup>. La comunidad Ryu es quien se encarga de efectuarle mejoras, solucionando los posibles fallos y agregando nuevas funcionalidades.

Mediante este controlador, se programan los conmutadores SDN que componen la conexión WAN privada del proveedor B de la *Figura 7*. Con la ayuda de la API de dicho controlador es que fue posible desarrollar una aplicación que permita aplicar políticas de encaminamiento en las tablas de flujo de los conmutadores.

#### 4.3.1 Traffic Monitor

Sobre el controlador Ryu se ejecuta la aplicación *Traffic Monitor*, que permite obtener en tiempo real el estado de los enlaces de la red de una forma rápida y sencilla. Esta 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.

Para este trabajo fue necesario realizar algunas modificaciones para adaptarlo a lo que se necesitaba. En primer lugar, se vio que había información irrelevante para este trabajo como por ejemplo las columnas de bytes cursados o paquetes con errores. Estas líneas del código fueron comentadas ya que no se utilizan y así es más fácil manejar la

---

<sup>8</sup> <https://github.com/faucetsdn/ryu>

información crítica. 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. Este valor no se calcula sino que es impuesto cuando se genera el escenario mediante la herramienta Wondershaper explicada en el apartado 4.2.1. En el código del *Traffic Monitor* se definió una variable estática y esta es impresa en las tablas que se ven en la *Figura 14*. A efectos de simplificar el escenario se configuraron todos los enlaces de la red de conmutadores en 100 Mbps.

Una vez ejecutada por primera vez la aplicación *Traffic Monitor*, esta realiza un escaneo de los conmutadores 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 14*. A modo de ejemplo se puede ver que el switch 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 0x00000000A00, 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 2 - Extracto de configuración del CONM\_A

Link Port						
switch	3584	3328	3072	2816	2560	
datapath	port	port-speed(Kbps)	port-freebw (Kbps)	max-capacity(Kbps)	port-stat	link-stat
00000000000000d00	1	72.3	99927.7	100000	up	Live
00000000000000d00	2	72.1	99927.9	100000	up	Live
00000000000000a00	1	72.1	99927.9	100000	up	Live
00000000000000a00	2	72.0	99928.0	100000	up	Live
00000000000000e00	1	72.1	99927.9	100000	up	Live
00000000000000e00	2	71.9	99928.1	100000	up	Live
00000000000000c00	1	72.1	99927.9	100000	up	Live
00000000000000c00	2	71.9	99928.1	100000	up	Live
00000000000000b00	1	72.0	99928.0	100000	up	Live
00000000000000b00	2	72.3	99927.7	100000	up	Live

Figura 14 – Tablas de topología y puertos por tráfico del programa Traffic Monitor de Ryu

Las columnas más relevantes de la *Figura 14* 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. Esta información es propia de la versión original de la aplicación de *Traffic Monitor* y se rigen mediante la siguiente ecuación:

$$\text{portFreebw (Kbps)} = \text{maxCapacity (Kbps)} - \text{portSpeed(Kbps)}$$

Ecuación 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 14* 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.

Esta información será utilizada para poder determinar la cantidad de tráfico cursado por los enlaces en tiempo real y decidir si será o no necesario aplicar las políticas de tráfico sobre los commutadores. En caso de que el ancho de banda utilizado por un canal supere el umbral definido previamente por el usuario, entonces se activarán automáticamente las políticas de encaminamiento sobre los commutadores. De este modo se enviará todo el tráfico que no requiera prestaciones de calidad por otro camino dejando lugar para enviar el tráfico que requiera mayores prestaciones por el camino corto.

#### 4.3.2 Ofctl\_rest

La interfaz northbound del commutador Ryu se implementa como una API REST (API ofctl\_rest [10]). 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 commutadores. La *Figura 15* representa una captura con Wireshark de la interfaz de loopback de la máquina virtual donde se muestra la secuencia que se genera al intentar modificar una entrada de la tabla de flujos de un commutador. La elección de la interfaz se debe a que es en esta donde se encuentra corriendo en controlador (127.0.0.1:6633).

- 1) El usuario solicita por ejemplo agregar una entrada de flujo en un commutador. Para ello establece una conexión TCP hacia el puerto 8080 donde se encuentran escuchando la API. Estos intercambian una serie de paquetes de sincronización TCP para luego realizar 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<sup>9</sup> la acción hacia el puerto TCP 6633 que tiene escuchando hacia el controlador.
- 3) El commutador modifica su tabla de flujo con la entrada configurada en el punto 1.

TCP	94 48562 → 8080 [SYN] Seq=0 Win=43690 Len=0 MSS=65476 SACK_PERM=1 TSval=1479546241 TSecr=0 WS=128
TCP	74 8080 → 48562 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
TCP	74 53918 → 8080 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=2341764260 TSecr=0 WS=128
TCP	74 8080 → 53918 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=2341764260 TSecr=2341764260 WS=128
TCP	66 53918 → 8080 [ACK] Seq=1 Ack=1 Win=43776 Len=0 TSval=2341764260 TSecr=2341764260
HTTP	510 POST /stats/flowentry/add HTTP/1.1 (application/x-www-form-urlencoded)
TCP	66 8080 → 53918 [ACK] Seq=1 Ack=445 Win=44800 Len=0 TSval=2341764260 TSecr=2341764260
HTTP	181 HTTP/1.1 200 OK
TCP	66 53918 → 8080 [ACK] Seq=445 Ack=116 Win=43776 Len=0 TSval=2341764262 TSecr=2341764262
TCP	66 53918 → 8080 [FIN, ACK] Seq=445 Ack=116 Win=43776 Len=0 TSval=2341764262 TSecr=2341764262
TCP	66 8080 → 53918 [FIN, ACK] Seq=116 Ack=446 Win=44800 Len=0 TSval=2341764262 TSecr=2341764262
TCP	66 53918 → 8080 [ACK] Seq=446 Ack=117 Win=43776 Len=0 TSval=2341764262 TSecr=2341764262
OpenFlow	162 Type: OFPT_FLOW_MOD

**Figura 15 – Captura de tráfico ante solicitud de modificación de entrada de flujo hecha con Wireshark**

<sup>9</sup> Tipo de paquete OpenFlow que permite modificar el estado de un switch OpenFlow

Los POST a la API de Ryu se hacen mediante json como se indica en el siguiente ejemplo. Este perteneciente a un fragmento de código del script *Sensible\_traffic\_long\_path.sh*, el cual se explica a continuación.

```

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

```

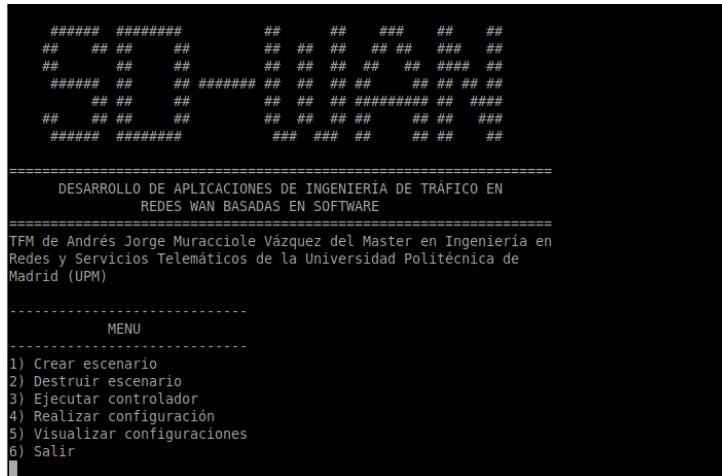
**Código 3 – Extracto de código de script “Sensible\_traffic\_long\_path.sh”**

La línea 2 del Código 3 indica el identificador del conmutador donde se desea aplicar la regla de tráfico. Seguido de esto, entre la 3 y la 7 se 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. Entre las líneas 8 y 13 se especifica el campo match de las tablas de flujo OpenFlow. Es decir, si las cabeceras del paquete que ingresa en el conmutador coinciden con los valores que se especifican en el match (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 refiere el campo “actions” (líneas 16 y 17). En este caso la acción consiste en despachar el paquete por el puerto 2 del conmutador. Por último, la línea 20 de este código indica la URL a la cual se hace el POST.

#### 4.4 Desarrollo de la aplicación SDN para ingeniería de tráfico

A continuación, se procede a explicar la aplicación SDN de ingeniería de tráfico que se ha implementado en este TFM. El uso de esta aplicación se encuentra totalmente automatizado. Es decir, se ha desarrollado un controlador SD-WAN sencillo para simular la gestión del servicio SD-WAN desde un punto central y con una interfaz “amigable” (*Figura 16*) que abstraiga al administrador de los detalles de las capas inferiores de la arquitectura SDN. El controlador SD-WAN se arranca con un ejecutable de escritorio que se ha nombrado TE\_SD-WAN. Tras pulsar en el ejecutable, se despliega un menú mediante el cual el administrador interactúa con él y a medida que éste va pulsando las opciones, se van ejecutando una serie de códigos (escritos en Bash o Python según corresponda) de forma transparente ante la vista del operario. Estos se van a explicar a lo largo de este capítulo.

Todos los códigos desarrollados para este TFM [10] se encuentran disponible de forma pública en el repositorio Github del autor <sup>10</sup>.

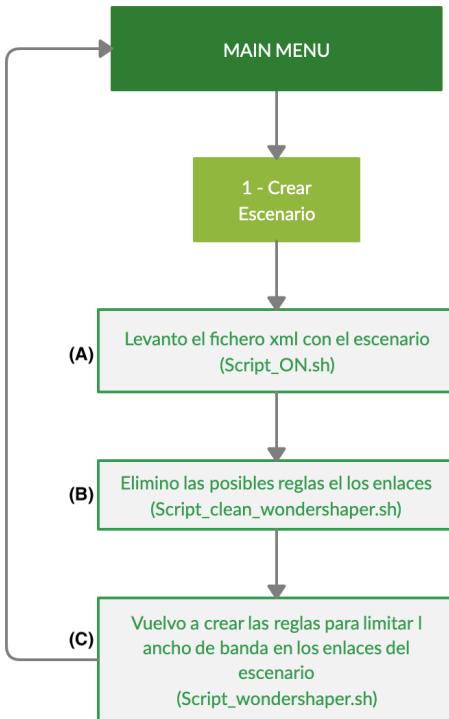


**Figura 16 – Menú principal interactivo**

#### 4.4.1 OPCIÓN 1: Crear escenario

Este paso es el primero que se debe realizar. Aquí lo que se hace es levantar (caja A) y dejar pronto el escenario sobre el cual luego se trabajará. Con él se instancian los contenedores que ofician de equipos finales, commutadores y SD-WAN Edges y se realizan las conexiones entre estos para crear la topología mencionada en el apartado 4.2. A su vez se ponen en funcionamiento las cajas B y C, las cuales mediante la herramienta Wondershaper se definen los anchos de banda de los canales de la red de commutadores en 100 Mbps de uplink y downlink. La secuencia de estos pasos se ven reflejados en el diagrama de la *Figura 17*.

<sup>10</sup> [https://github.com/amuracciole/TrafficEngineering\\_SDWAN.git](https://github.com/amuracciole/TrafficEngineering_SDWAN.git)



**Figura 17 – Diagrama. OPCIÓN 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.

En este punto aún no se ha encendido el controlador Ryu, con lo cual hasta que no se haga, no habrá conectividad entre los hosts ya que los commutadores desconocen como encaminar el tráfico. La puesta en funcionamiento el controlador se menciona más adelante en el apartado 4.4.3.

#### 4.4.2 OPCIÓN 2: Destruir escenario

La opción número 2 realiza la secuencia que se ilustra el diagrama de la *Figura 18*, la cual destruye el escenario creado en el apartado 4.4.1.

Cuando se selecciona la opción dos, se ejecuta el 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 realizan modificaciones en el fichero de configuración del escenario (fichero XML).

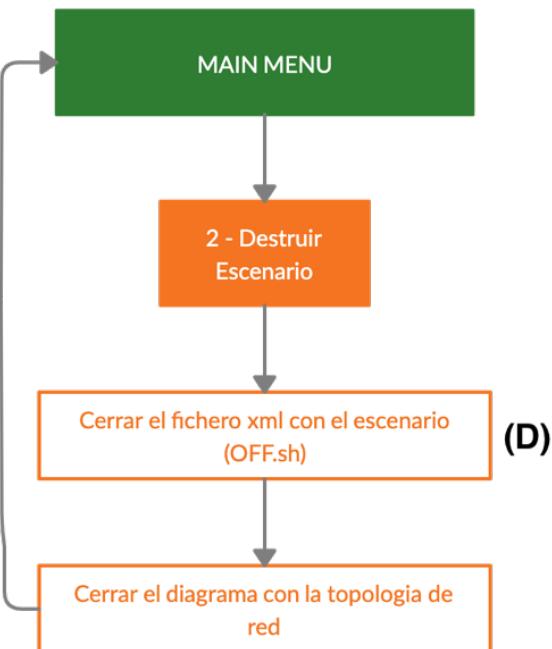


Figura 18 – Diagrama. OPCION 2) Destruir Escenario

#### 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 (caja F) e IP tanto versión 4 como versión 6 (caja G) y llamar al script *Monitor.py* (caja H) como se ve en el diagrama de flujo de la *Figura 19*. Este último se encuentra constantemente censando los enlaces de los commutadores.

En la caja E se inicializa el controlador Ryu con el *Traffic Monitor* adaptado, el cual permite por un lado conectar a él los commutadores y por otro, imprimir cada 10 segundos el estado de la red en consola. A su vez, cada vez que actualiza las tablas recientemente mencionadas, también abre un fichero csv, guarda dichos valores allí y lo cierra. Esta información guardada en este archivo de texto será de vital importancia para que luego el programa encargado de realizar la ingeniería de tráfico (*Monitor.py*) pueda trabajar con ella.

Al iniciar por primera vez 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 necesario forzar manualmente un camino por defecto por el cual enviar el tráfico entre los SD-WAN Edge de la red. Aquí entran en juego las cajas F y G las cuales son Shell scripts formados por una serie de POST de json a la API (similares al puesto como ejemplo en el *Código 2*) que permiten configurar el controlador por la interfaz northbound. Estos hacen que Ryu escriba las entradas de flujo necesarias en los commutadores CONM\_A, CONM\_C y CONM\_E y así permitir enrutar todo el

tráfico IPv4, IPv6 y ARP por un camino. De esta forma se evitan los bucles de capa 2 ya que se define un único camino para dicho tráfico. Esta configuración se realiza por defecto para que envíe todo el tráfico por el camino corto, pero luego se verá que se puede cambiar tanto de forma manual como automatizada.

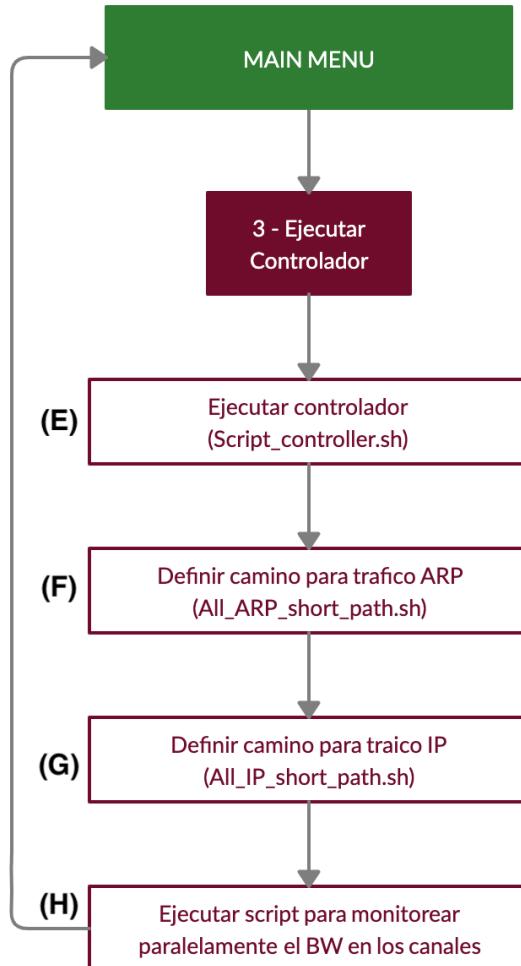


Figura 19 – Diagrama. OPCIÓN 3) Ejecutar Controlador

El script *Monitor.py* (caja H) de la Figura 19 es quien ejecuta la aplicación SDN de ingeniería de tráfico a partir de los datos recopilados por el Traffic Monitor. Una vez que el monitor es inicializado, éste permanece funcionando de forma paralela al controlador pero consumiendo datos que va recopilando y guardando de forma periódica en un archivo .csv tal cual se explicó en el punto 4.3.1.

Cada 10 segundos el *Traffic Monitor* reescribe el archivo csv con los valores de anchos de banda disponible en cada enlace. Antes de que estos cambien, el *Monitor.py* los copia a dos archivos de texto llamados *Data\_long.sh* y *Data\_short.sh*, los cuales van guardando la serie histórica de los datos de ancho de banda de cada enlace según corresponda. *Data\_long.sh* almacena una tabla de cuatro columnas donde en las primeras dos aparecen reflejados el tiempo ( períodos de 10 segundos) y el ancho de banda libre del *long\_path*

respectivamente. La tercera columna guarda el valor del umbral máximo admitido para el tráfico sensible (en este caso 65000 en referencia a 65 Mbps) y en la cuarta el valor constante de 100000 que refiere a la capacidad máxima del canal. Estas dos ultimas son constantes ya que obtienen y guardan el dato definido en una variable del código pero que son importantes luego para poder graficar y ver mas fácilmente cuan cerca de la cota máxima o de la capacidad del canal se está. Por otra parte, el *Data\_short.sh* es muy similar al anterior solo que la segunda columna almacena el ancho de banda disponible del enlace *short\_path*. A su vez, el programa *Monitor.py* calcula el promedio porcentual de enlace disponible en los últimos 30 segundos y si este promedio supera el umbral crítico definido (< 65% utilizado) entonces activa las políticas que modifican las tablas de flujo para el tratamiento del tráfico sensible. Se decidió analizar el promedio en los últimos 30 segundos para generar mas estabilidad al sistema. Es decir, si sube del 65% pero a los 10 segundos baja, no es necesario aplicar las políticas. Este promedio es un intervalo de confianza y hace que los cambios en las tablas de flujos sean mas suaves. Si bien en este trabajo las simulaciones se hacen con tasas de tráfico constantes como se verá mas adelante; en un escenario real el tráfico suele ser cambiante.

Por lo tanto, si en los últimos 30 segundos se aprecia un uso del canal promedio **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*). Si bien este tráfico es insignificante en cuanto al consumo de ancho de banda, la idea de moverlo hacia el otro camino es debido a que se busca dejar lo mas libre posible el *short\_path* para el tráfico sensible. Por otro lado, es necesario moverlo y no eliminar este tipo de tráfico ya que no se quiere que existan cortes de servicio.
- 3) Ejecutar un script que envía el tráfico IPv4 no sensible por el camino largo (*long\_path*)

Si en algún momento la disponibilidad del camino 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 20*.

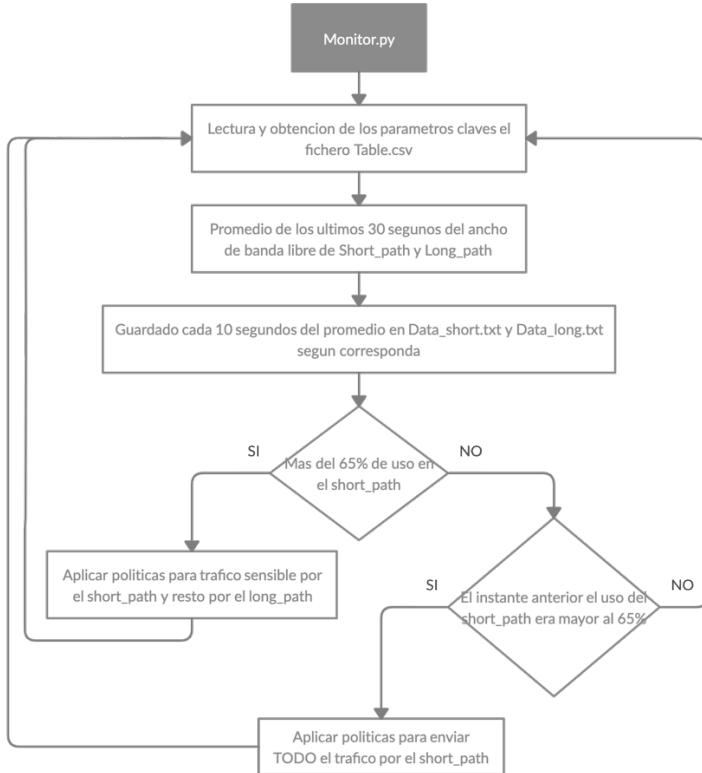


Figura 20 – Diagrama Monitor.py

La forma por la cual se mide el ancho de banda de los caminos es a partir del valor de ancho de banda de las interfaces 1 y 2 del CONM\_A, las cuales están conectados a los conmutadores B y C respectivamente. Esta dedición se tomó por un lado teniendo en cuenta que por como está configurado el escenario, no habrá tráfico terminado en los conmutadores. Todo el tráfico se origina y termina en los hosts o en su defecto en los SD-WAN Edge. Por tal motivo es correcto asumir que el ancho de banda disponible en el enlace que conecta los conmutadores A y B, es igual al que existe entre B y D y entre D y E. De la misma forma, el ancho de banda libre en el enlace que conecta los conmutadores A y C es igual al que conecta el C con el E. A raíz de ello es que solamente se mide el ancho de banda disponible en los enlaces hacia dentro de la red de conmutadores del CONM\_A y se asume que este valor representa correctamente el ancho de banda libre de todo el camino.

El valor del umbral crítico anteriormente mencionado fue elegido de forma arbitraria para facilitar las prácticas y la muestra de la solución. De todas formas este valor es fácilmente modificable desde la variable “bw\_lim” del `Monitor.py`.

Con esto se logra llegar a una solución que permite que mientras el camino formado por los conmutadores A,C y E esté relativamente libre (menos del umbral crítico) todo el tráfico viaja por él. Sin embargo, cuando éste supere el umbral crítico de uso, todo el tráfico que no tenga unos requisitos estrictos de QoS se envía por el camino mas largo,

dejando el corto solo para el tráfico “sensible”. 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 programa de monitoreo de enlaces se encuentra constantemente imprimiendo en una consola el porcentaje de enlace libre como muestra la *Figura 21*. De esta forma el administrador de la red puede conocer su estado en tiempo real, aunque también lo va a poder consultar gráficamente como se muestra en el punto 4.4.5.

```
LONG_PATH (CONN_A - CONN_B - CONN_D - CONN_E):
    Bandwidth: 1.93 Kbps
    % libre (Prom. ultimos 30 seg.): 99.9981%
()
SHORT_PATH (CONN_A - CONN_C - CONN_E):
    Bandwidth: 1.93 Kbps
    % libre (Prom. ultimos 30 seg.): 99.9981%
```

**Figura 21 – 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 configuraciones IP (1), agregar configuración ARP (2) o eliminar configuraciones IP (3) tal cual se muestra en la *Figura 22*. Estas opciones se explican a continuación. A su vez todo el proceso se detalla en el diagrama de la *Figura 23*.



**Figura 22 – Menú de las posibles configuraciones**

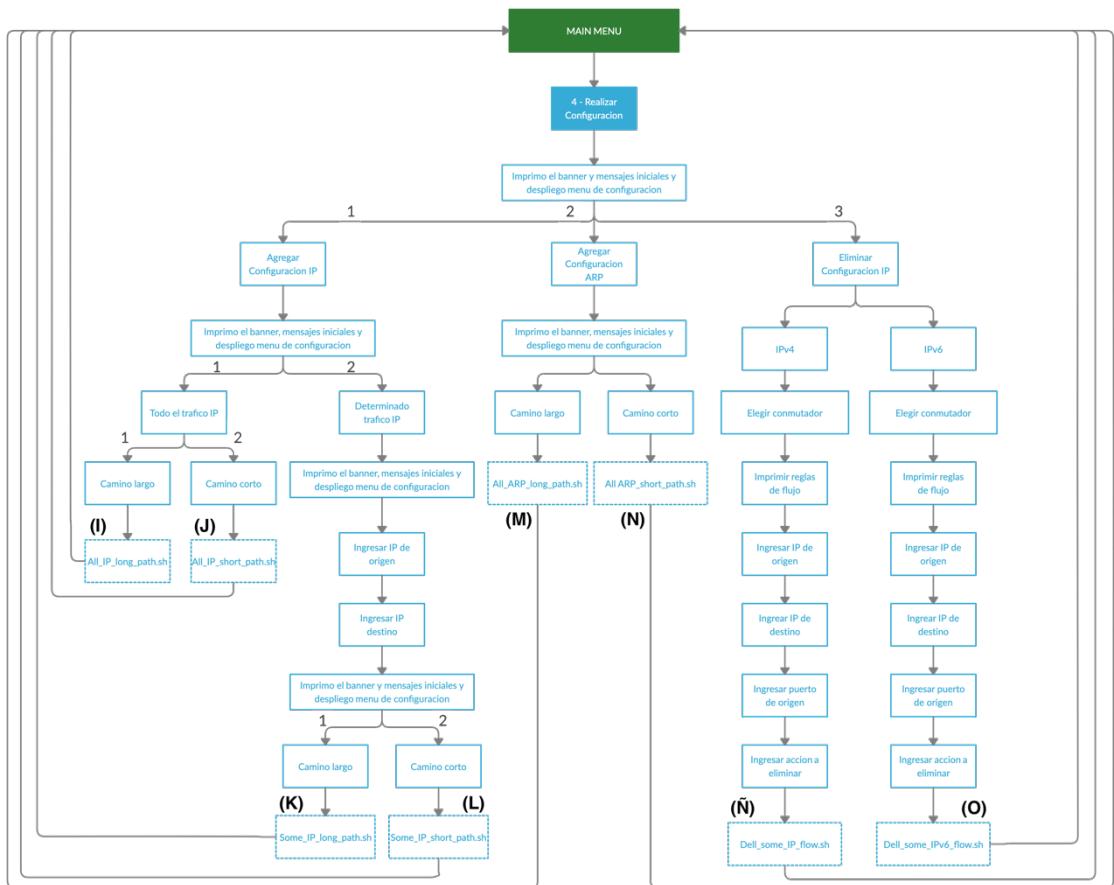


Figura 23 – Diagrama. OPCION 4) Realizar Configuración

### 1. Agregar configuración IP

Dentro de esta opción, el operador puede configurar de forma genérica el camino que deben tomar todos los paquetes IPv4 e Ipv6, sin importar el origen o destino. Esto se hace desde la sub-opción 1. A su vez, con la sub-opción 2 se puede configurar el camino de un determinado flujo Ipv4 o Ipv6, es decir, elegir un camino específico para el flujo con una dirección IP de origen y destino particular. Introducir una opción incorrecta le permite cancelar la operación y volver a la pantalla inicial.

En caso de optar por la primer opción, luego es necesario seleccionar explícitamente el camino. Si se selecciona el camino largo se ejecuta “All\_IP\_long\_path.sh” (caja I). Si se selecciona el camino corto se ejecuta “All\_IP\_short\_path” (caja J).

De tomar la sub-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

operador de forma manual, entonces siempre se va a cumplir sin importar si la solución TE intenta direccionar ese tráfico por otro camino. Esto se debe a que las configuraciones 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**.

El motivo por el cual se decidió que las opciones que envían todo el tráfico IP y ARP por un camino sean configuradas con prioridad 0 y no con 10 como es el caso del resto de las configuraciones manuales es que en caso de que se hagan con otro valor distinto de 0, una configuración así dejaría sin efecto la solución TE de este TFM. Para poder hacer que estas reglas se creen con prioridad 10, es necesario realizar modificaciones dentro de la opción “Eliminar configuración IP” del punto 3 de este capítulo. Esto escapa a este TFM y no aporta mayores beneficios respecto al objetivo del trabajo.

## **2. Agregar configuración ARP**

Aquí solo se permite elegir el camino que debe tomar todo el tráfico ARP (caja M o N). En caso de no definir un camino específico para el tráfico ARP, habrá un loop en la red de conmutadores ya que se enviarán constantemente ARP request y solicitations por ambos caminos de la red. Esto generaría una congestión de la red debido a los mensajes ARP de broadcast, llegando a saturar en los enlaces en pocos segundos y volviendo así inoperable la red.

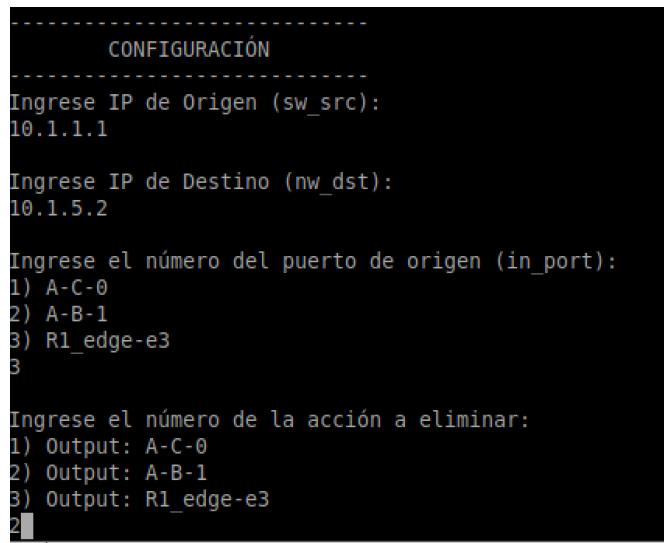
*Tanto para el tráfico IP como ARP solo es necesario configurar el camino de ida ya que la vuelta se configura automáticamente de forma análoga. Esto es gracias a que los scripts están configurados en ambos sentidos. Es decir, tanto cuando se configura a nivel general como cuando se introduce la información con las direcciones de origen X.X.X.X y destino Y.Y.Y.Y, automáticamente se crean los POST con esta información y también con origen Y.Y.Y.Y y destino X.X.X.X. A modo de simplificar el trabajo del operador, este solo debe introducir la información en una única dirección. A la vuelta los paquetes tomarán el mismo camino en sentido contrario.*

## **3. Eliminar configuración IP**

Mientras que la sub-opción 2 dentro de “Agregar configuración IP” permite añadir entradas en las de flujo de los conmutadores especificando las direcciones IP de origen y destino del tráfico, la opción “Eliminar configuración IP” permite eliminarla. 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 entrada mas prioritaria y por ende la configuración será permanente.

Una vez introducida la opción de eliminar la configuración, es necesario elegir primero el protocolo IP de la regla a borrar. Una vez hecho esto se elige el conmutador

en donde existe 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. Por la forma en como está hecha la solución, para borrar una entrada el operador debe ir 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 N o O según corresponda). Esta secuencia se puede ver en la *Figura 24*.



```
CONFIGURACIÓN
-----
Ingrese IP de Origen (sw_src):
10.1.1.1

Ingrese IP de Destino (nw_dst):
10.1.5.2

Ingrese el número del puerto de origen (in_port):
1) A-C-0
2) A-B-1
3) R1_edge-e3
3

Ingrese el número de la acción a eliminar:
1) Output: A-C-0
2) Output: A-B-1
3) Output: R1_edge-e3
2
```

**Figura 24 - Menú interactivo para eliminar configuración IPv4**

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. El motivo por el cual solo se pueden eliminar estas entradas y no por ejemplo las creadas por la aplicación de ingeniería de tráfico es principalmente por dos razones. En primer lugar no tendría sentido ya que como éstas se crean en función de la demanda, a pesar aunque se borren manualmente, estas reglas se volverían a crear una vez de vuelva a superar el umbral. El operador debería estar borrando manualmente las entradas constantemente lo cual no tiene sentido. En segundo lugar, haciéndolo de esta forma evita posibles fallos del operador y da mas seguridad a la red ya que en caso de borrarse por ejemplo una entrada creada por las solución TE, esta dejará de funcionar correctamente pudiendo traer problemas en la red.

#### 4.4.5 OPCIÓN 5: Visualizar configuraciones

Esta opción se ha definido para poder visualizar la programación de los conmutadores de la red. El menú correspondiente a este punto se muestra en la *Figura 25* y se pasa a detallar cada una de las opciones en los siguientes apartados.



Figura 25 – Menú de las posibles visualizaciones

La Figura 26 corresponde al diagrama de flujo de todo lo referido a la visualización de configuraciones.

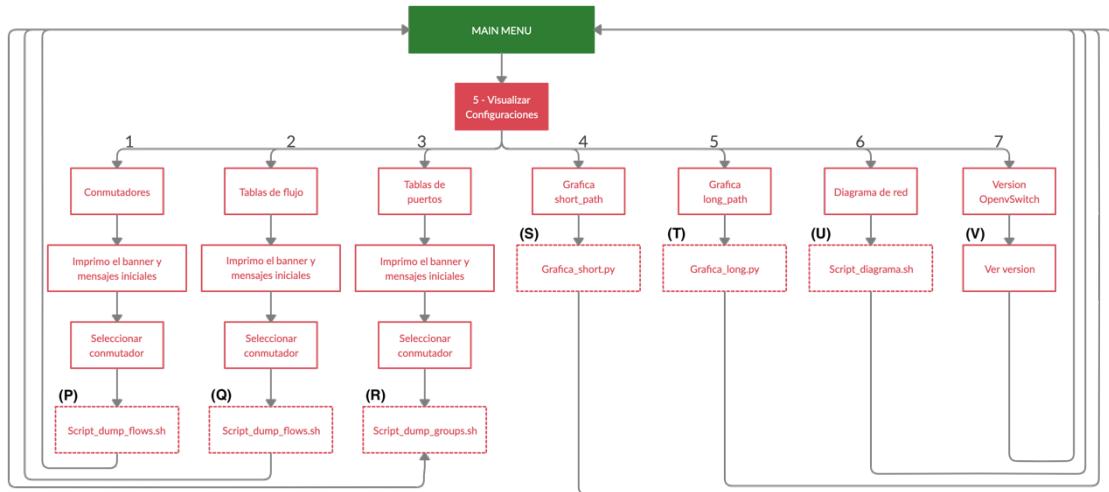


Figura 26 – Diagrama. OPCIÓN 5) Visualizar Configuraciones

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

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

El comando para esta visualización es el siguiente: “*ovs-vsctl show*” (caja P).

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

Mediante esta opción es posible ver las tablas de flujos de los commutadores y así entender qué comportamiento está asumiendo cada uno a la hora de redirigir los paquetes. A modo de ejemplo se muestra la Figura 27 la cual corresponde a la tabla del CONM\_A. Los campos que componen cada entrada se detallan a continuación [11].

```

cookie=0x0, duration=3329.087s, table=0, n_packets=6932, n_bytes=415920, priority=65535,dl,dst=0:80:c2:00:00:0e,dl_type=0x88cc actions=CONTROLLER:65535
cookie=0x0, duration=9.822s, table=0, n_packets=0, n_bytes=0, send flow rem priority=10,ip,in port="R1 edge-e3",nw_src=10.1.1.2,nw_dst=10.1.1.2 actions=output:"A-C-0"
cookie=0x0, duration=328.146s, table=0, n_packets=0, n_bytes=0, send flow rem priority=10,ip,in port="A-C-0",nw_src=10.1.5.2,nw_dst=10.1.1.2 actions=output:"R1_edge-e3"
cookie=0x0, duration=328.146s, table=0, n_packets=0, n_bytes=0, send flow rem priority=1,arp,in port="R1 edge-e3" actions=output:"A-C-0"
cookie=0x0, duration=328.129s, table=0, n_packets=0, n_bytes=0, send flow rem priority=1,arp,in port="A-C-0" actions=output:"A-C-0"
cookie=0x0, duration=328.033s, table=0, n_packets=0, n_bytes=0, send flow rem priority=1,ip,in port="A-C-0" actions=output:"R1_edge-e3"
cookie=0x0, duration=328.033s, table=0, n_packets=0, n_bytes=0, send flow rem priority=1,ip,in port="A-C-0" actions=output:"A-B-1"
cookie=0x0, duration=314.693s, table=0, n_packets=0, n_bytes=0, send flow rem priority=1,ip,in port="A-B-1" actions=output:"R1_edge-e3"
cookie=0x0, duration=329.087s, table=0, n_packets=21, n_bytes=1470, priority=0 actions=CONTROLLER:65535

```

**Figura 27 – 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 commutados a partir de dicha entrada de flujo.

n\_bytes: Cantidad de bytes que fueron commutados a partir de dicha entrada de flujo.

priority: Valor de la prioridad. Ante dos posibles entradas coincidentes, el controlador elige commutarlo 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.

El comando Ovs-ofctl para ver las tablas de flujo es el siguiente: “*ovs-ofctl -O OpenFlow13 dump-flows X*”, siendo X el nombre del conmutador (caja Q). Por ejemplo CONM\_A.

### **3. Tabla de puertos (ports\_tables)**

Permite ver información más detallada de los puertos de los conmutadores. Además de la información obtenida de las entradas de flujo de la Figura 27, se puede obtener información mas detallada como la cantidad de paquetes descartados, paquetes con errores, etc.

El comando Ovs-ofctl para ver las tablas de puertos es el siguiente: “*ovs-ofctl -O OpenFlow13 dump-ports X*”, siendo X el nombre del conmutador (caja R). Por ejemplo CONM\_A.

#### 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 ancho de banda consumido en los diferentes caminos de la conexión WAN privada (en Mbps). Las Figuras 28 (a) y (b) muestran ejemplos de ello.

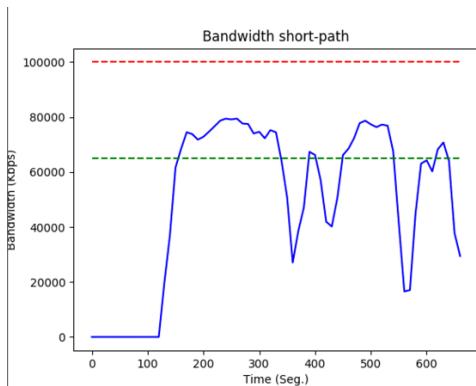


Figura 28 (a) Gráfica de Bandwidth (Kbps) en función del tiempo (segundos) del short\_path

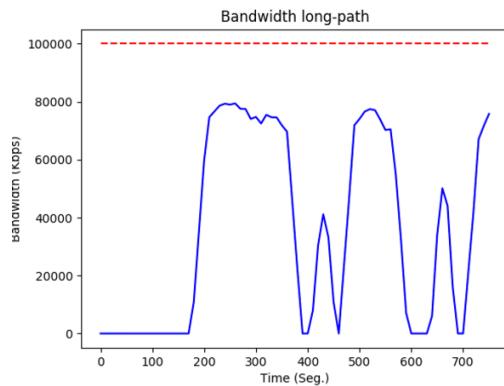


Figura 28 (b) Gráfica de Bandwidth (Kbps) en función del tiempo (segundos) del long\_path

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 (Data\_long.txt y Data\_short.txt respectivamente). 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 S) y *Graph\_long.py* (caja T) respectivamente los cuales obtienen los valores de sus archivos de texto y los grafican en pantalla. Para su programación fue necesario utilizar las librerías *numpy* que permiten trabajar los datos de forma matricial y las librerías *matplotlib* y *time* para permitir graficar.

Las gráficas son interactivas con el usuario de forma que se permite hacer zoom en cualquier parte del gráfico, 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 11 vista anteriormente.

El comando Ovs-ofctl para ver la topología de red del escenario virtual es el siguiente: “*sudo vnx -f /root/NAME.xml --show-map*”, siendo root la ruta donde está el fichero xml del escenario guardado y NAME el nombre del archivo .xml donde se encuentra la configuración del escenario (caja U).

## 7. Versión OpenvSwitch

La finalidad de esta opción es únicamente devolver la versión de los OVS's que forman el escenario virtual (caja V).

El comando Ovs-ofctl para ello es el siguiente: "*ovs-vsctl -version*".

### 4.4.6 OPCIÓN 6: Salir

La última opción del menú principal corresponde a "salir". Una vez elegida, se ejecuta un script que 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, cierra el controlador Ryu y la ventana con el diagrama del 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 proporcionarle, por parte del administrador de la red, una prioridad respecto al resto. Con la ayuda del documento "SD-WAN Quality of service" de Aruba [12] se determinó que los siguientes protocolos serán los designados como ejemplos de tráfico 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 2 – Protocolos catalogados como tráfico sensible

Se tratarán todos los protocolos de la Tabla 2 de igual forma a modo de simplificar el funcionamiento de la solución aunque en un escenario real es factible pensar que dentro del tráfico sensible, hay algunos que lo son más respecto a otros.

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 puertos de la Tabla 2 para simular estos tráficos.

PUERTO TCP REAL	PUERTO TCP FICTION	PUERTO UDP REAL	PUERTO UDP FICTION
5060	35000	5060	35000
22	35001	500 y 4500	35001 y 35002

Tabla 3 – Mapeo de puertos para utilizar en iperf

Estos puertos mapeados son los que luego se configuran en el *Monitor.py* para que sean tratados como prioritarios y son los utilizados en las simulaciones hechas en el capítulo 5 de este documento.

## 5 Pruebas y resultados obtenidos

Para validar el comportamiento de la aplicación SDN se han diseñado cinco experimentos. El experimento 1 consiste en demostrar el comportamiento que sufren las tablas de flujo de los commutadores cuando son modificadas de forma manual por el operador de la red. El experimento 2 buscará mostrar si ante cambios de configuración en el encaminamiento de paquetes, se aprecian pérdidas o si la red no se entera de estos cambios. Por otra parte, con los experimentos 3 y 4 se buscará mostrar y comparar el aporte de la ingeniería de tráfico diseñada en este TFM. En primer lugar se harán pruebas del comportamiento de la red sin TE y se estudiará la información referida a pérdida de paquetes, velocidades alcanzadas, jitter y uso de los enlaces. Luego, en el experimento 4 se volverán a hacer las mismas pruebas pero ahora con la ingeniería de tráfico funcionando. Se volverán a medir estas variables y se las comparará con las del experimento 3. Para ambos casos de inyectará tráfico TCP a 60 Mbps (por debajo el umbral crítico) que oficiará de tráfico no prioritario y tráfico UDP que simulará tráfico sensible. De esta forma se busca saturar los enlaces y ver cómo compiten por ellos. Por último, el experimento 5 consiste en una prueba similar a la de los experimentos 3 y 4 pero ahora el tráfico no sensible inyectado será superior al umbral crítico (100 Mbps). Con esto no se deja prácticamente lugar para enviar paquetes UDP (catalogados como sensibles). De esta forma se busca realizar pruebas de stress en la red. Aquí se volverá a comparar los resultados obtenidos con y sin solución TE para ver su funcionamiento ante grandes cantidades de tráfico en la red SD-WAN.

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

En este experimento se buscará mostrar lo rápido e intuitivo que resulta modificar el escenario utilizando la interfaz del controlador SD-WAN. Se mostrará que es posible modificar el camino que toma el tráfico para ir de un origen a un destino cuando el operador lo deseé sin tener que detener el servicio.

En primer lugar se levanta el escenario y el controlador seleccionando las opciones 1 y 3 que se listan en el menú del controlador SD-WAN. Como se mencionó anteriormente, una vez que se ejecuta el escenario por primera vez, este define el camino corto como por defecto, creando las entradas de flujo en los commutadores. En la *Figura 29* se muestra la tabla de flujo de CONM\_A donde se ve en la línea 4 que el tráfico IP proveniente el router R1 se conmuta hacia el CONM\_C (A-C-0).

```

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 29 – Tabla de flujo del CONM\_A por defecto**

Haciendo un TCP iperf<sup>11</sup> de 50 Mbps entre h01 y h03 se ve efectivamente, como puede observar en las *Figuras 30 (a)* y *(b)* que todo el tráfico va por el short\_path.

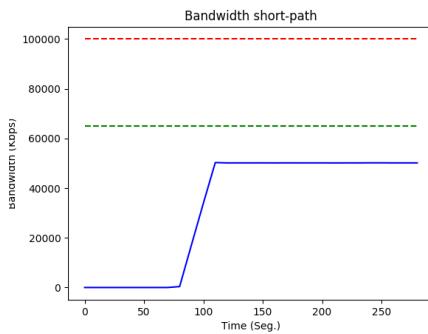


Figura 30 (a) Exp. 1. Ancho de banda utilizado del short\_path en función del tiempo

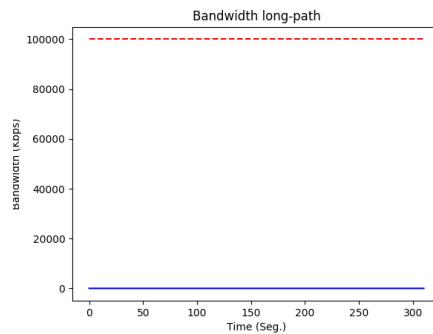


Figura 30 (b) Exp. 1. Ancho de banda utilizado del long\_path en función del tiempo

Sin detener el iperf se modifica la ruta que sigue el tráfico desde el controlador SD-WAN de forma que ahora todo el tráfico IP se encamine por el long\_path. La programación de los commutadores se puede verificar observando tanto la línea 5 de la *Figura 31* 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 las *Figuras 32 (a)* y *(b)*. De estas últimas se puede ver que una vez hecho el cambio de configuración, el tráfico comenzó a ir por el long\_path.

<sup>11</sup> **Servidor:** “iperf -s -i 1 -t 1000 -b 50000000”. **Cliente:** “iperf -c 10.1.5.2 -i 1 -t 1000 -b 50000000”

```

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 31 – Tabla de flujo del CONM\_A luego de cambiar la configuración de ruteo**

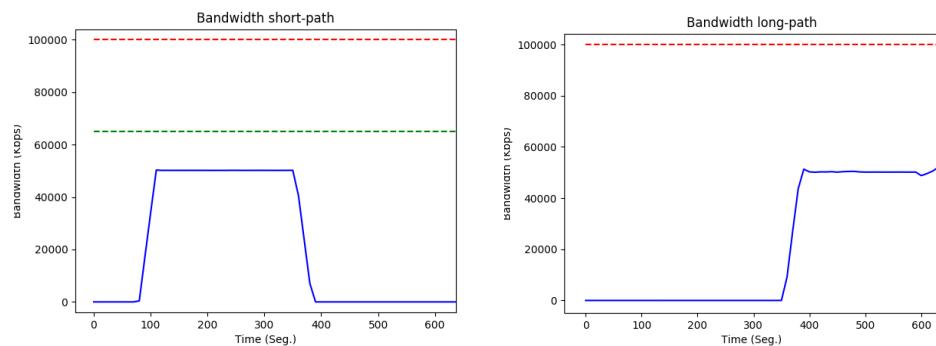


Figura 32 (a) Exp. 1. Ancho de banda utilizado del short\_path en función del tiempo después de la configuración de caminos

Figura 32 (b) 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 commutadores se hace de forma correcta e instantánea. Por otra parte se desprende la simpleza con la que 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.

## 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<sup>12</sup> 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 las Figuras 33 (a) y (b). Una vez transcurrido el tiempo se obtienen los siguientes resultados:

<sup>12</sup> Servidor: "iperf -s -i 1 -t 500 -u". Cliente: "iperf -c 10.1.5.2 -i 1 -t 500 -u"

- Paquetes enviados: 44583
- Paquetes perdidos: 10 (0,022%)
- Jitter promedio: 0,044 ms

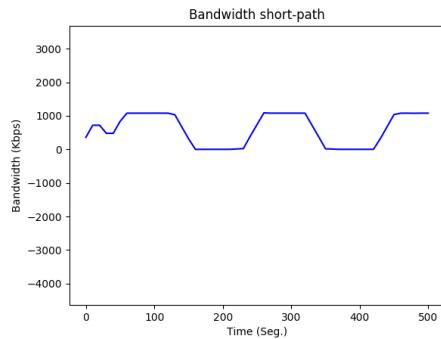


Figura 33 (a) Exp. 2. Ancho de banda utilizado del short\_path en función del tiempo

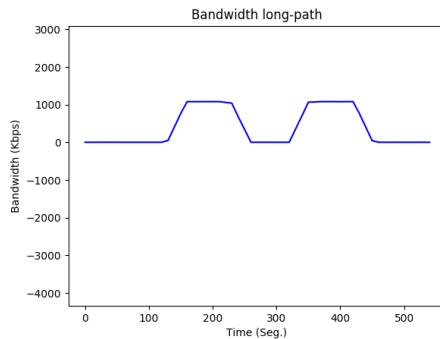


Figura 33 (b) Exp. 2. Ancho de banda utilizado del long\_path en función del tiempo

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%)

Tabla 4 – Resultados de iperf UDP cada 100 segundos

De esta forma se puede ver que todos los paquetes perdidos fueron resultado del primer cambio de configuración, pero en las siguientes, la pérdida fue de 0%. El motivo por el cual sucedió esto es que en el primer cambio fue necesario crear una entrada nueva en los commutadores pero en las demás lo único que se hizo fue una modificación de las mismas lo que redujo el tiempo de respuesta. Esto se puede apreciar también si se comparan la *Figura 29* y la *Figura 31*. 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 ser 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 tienen como objetivo mostrar la importancia de una solución de ingeniería de tráfico en los entornos SD-WAN. Las pruebas son las mismas en ambos casos con la diferencia de que en el experimento 3 se deshabilita el programa *Monitor.py*, con lo cual no se hará re-direccionamiento del tráfico sensible a pesar de que pueda haber congestión en los enlaces del camino más corto.

Mediante la comparación de los resultados se busca poder determinar las ventajas que se obtienen al aplicar soluciones de TE en términos de pérdida y tasa de transmisión de paquetes.

Este experimento, al igual que el experimento 4, consiste en simular “tráfico sensible” UDP (por ejemplo voz sobre IP) a 60 Mbps entre h01 y h03<sup>13</sup> y tráfico no prioritario TCP entre h02 y h04<sup>14</sup>. Por la definición de estos tráficos lo que sucede es que el tráfico TCP consumirá los 60 Mbps y el UDP el resto disponible. En esta prueba se deshabilita la aplicación de TE para analizar cómo compiten los flujos por el ancho de banda disponible.

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

- 1) Todo el tráfico (UDP y TCP) fue por el camino corto (*short\_path*) y nada por el largo tal cual se puede ver en las *Figuras 34 (a)* y *(b)* respectivamente.

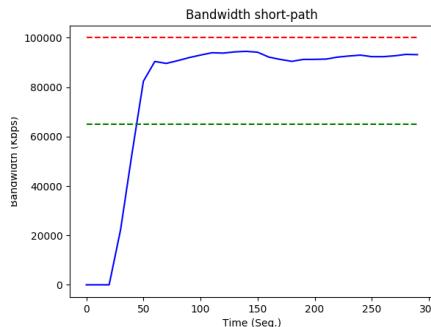


Figura 34 (a) Exp. 3. Ancho de banda utilizado del *short\_path* en función del tiempo sin TE

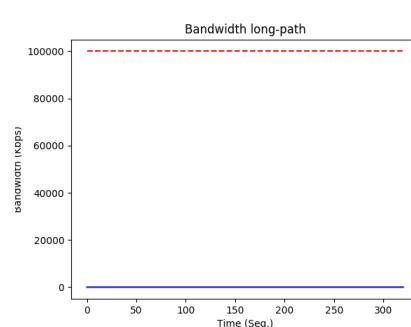


Figura 34 (b) Exp. 3. Ancho de banda utilizado del *long\_path* en función del tiempo sin TE

- 2) No solo no se tuvo en cuenta la categorización de tráfico sensible, lo cual es obvio ya que la aplicación de TE estaba deshabilitada, sino que al ser del tipo UDP frente a uno TCP, este tuvo pérdidas. Mas precisamente a partir de las estadísticas obtenidas con el iperf, el “tráfico sensible” (UDP) tuvo unas pérdidas del 25% (376960 paquetes en un total de 1530613).

<sup>13</sup> **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”

<sup>14</sup> **Servidor:** “iperf -s -i 1 -t 300”. **Cliente:** “iperf -c 10.1.5.3 -i 1 -t 300”

En definitiva se ve que no hubo un uso óptimo de la red ya que en primer lugar el tráfico VOIP que requería de ciertos estándares de calidad de servicio se vio afectado por otros tráficos no prioritarios. Por otro lado, el enlace long\_path nunca se utilizó a pesar de que el otro estaba saturado.

#### 5.4 Experimento 4: Encaminamiento de tráfico sensible mediante soluciones de TE

En este experimento 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 enlace, este realizará configuraciones en el controlador para intentar optimizar el tráfico. Recordar que este valor de umbral fue definido de forma arbitraria únicamente para las pruebas pero es posible modificarlo tal cual se explica en el apartado 4.4.3.

De esta forma dejará el tráfico UDP (el tráfico VOIP de la corporación) en el camino más corto, enviando el resto de tráfico (no sensible) por el camino más largo.

- 1) Se envió tráfico por ambos enlaces tal cual se puede ver en las *Figuras 35 (a)* y *(b)*, con lo cual se utilizó de una forma más eficiente la infraestructura de red existente respecto al experimento anterior. De estas gráficas también se desprende que esta es otra manera de resolver el problema del “pez” ya que el tráfico es balanceado sin la necesidad de modificar las métricas de los enlaces manualmente.

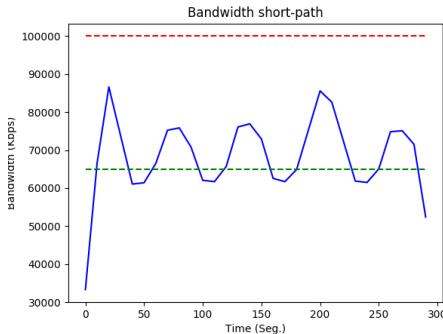


Figura 35 (a) Exp. 3. Ancho de banda utilizado del short\_path en función del tiempo con TE

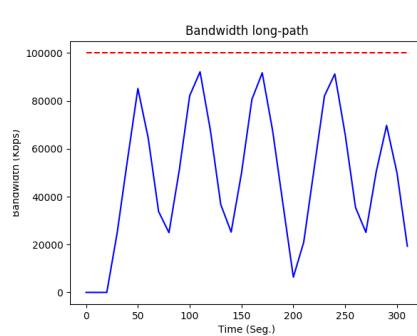


Figura 35 (b) Exp. 3. Ancho de banda utilizado del long\_path en función del tiempo con TE

- 2) Cuando se supera el umbral crítico, el programa lo detecta y aplica las políticas de tráfico que permiten enviar el tráfico no prioritario por un camino alternativo (el camino mas largo). A modo de ejemplo se puede ver que cerca del segundo 200, se activa la política en los comutadores haciendo que el tráfico por el short\_path comience a disminuir y crezca a el que viaja por el long\_path.
- 3) En este caso las pérdidas de tráfico UDP promedio durante los 300 segundos fue de 12% (184259 paquetes de 1530613). Si bien la pérdida disminuyó, esta sigue siendo considerable. Esto se debe a que mientras competían por el mismo enlace (tráfico sensible y tráfico no sensible) las pérdidas eran de entre 20 y 30%. Esto es coherente si se ven los resultados del experimento 3 ya que en ese momento los paquetes UDP solo pueden utilizar el ancho de banda que deba libre los paquetes

TCP. 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 0% como se puede ver en la secuencia de la *Figura 36*, la cual representa una captura de pantalla del comando iperf UDP entre h01 y h03.

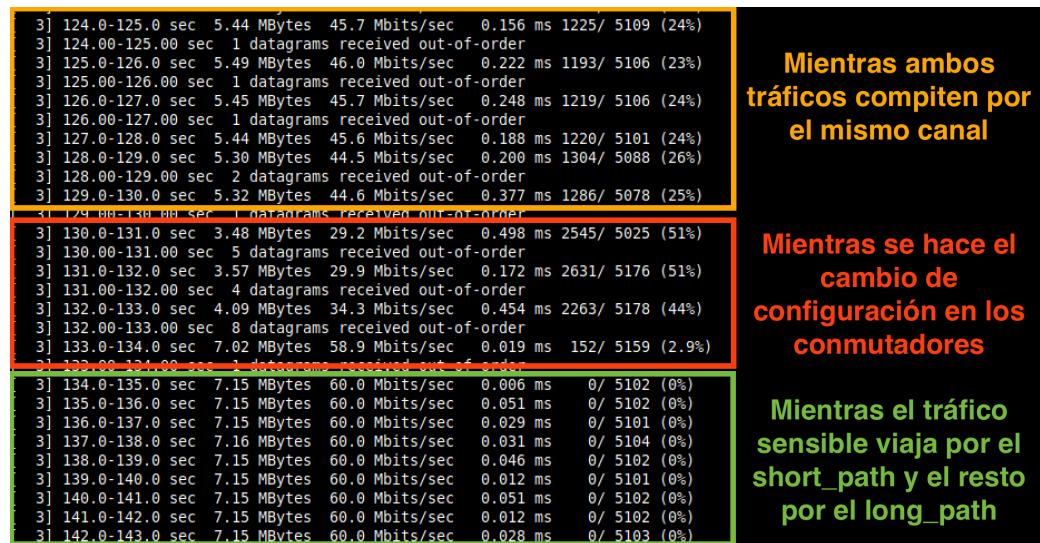


Figura 36 – Exp. 4. Pérdida de paquetes antes, durante y después de aplicar TE

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

Esta ultima prueba es muy similar a la realizada en los experimentos 3 y 4, con la diferencia de que ahora se simulará un tráfico sensible muy pesado (90 Mbps) el cual supere el umbral crítico. Con esto se busca comparar las pérdidas de paquetes obtenida sin TE y con TE y ver si ante estos casos de estrés la mejoría es considerable.

Deshabilitando la TE, se simula un tráfico sensible UDP<sup>15</sup> entre h01 y h03 de 90 Mbps y un tráfico no sensible TCP<sup>16</sup> 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* se plasman los resultados.

<sup>15</sup> Servidor: “iperf -s -i 1 -t 300 -p 35000 -u -b 90000000”. Cliente: “iperf -c 10.1.5.2 -i 1 -t 300 -p 35000 -u -b 90000000”

<sup>16</sup> Servidor: “iperf -s -i 1 -t 300”. Cliente: “iperf -c 10.1.5.3 -i 1 -t 300”

	<b>Sin TE</b>	<b>Con TE</b>
<b>Paquetes enviados</b>	2295920	2295862
<b>Paquetes perdidos</b>	1115705	180428
<b>Pérdida de paquetes (%)</b>	49	7,9
<b>Bandwidth UDP promedio (Mbps)</b>	46,3	82,9

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

En esta prueba se observa una mejora significativa al aplicar la solución de TE. Para el caso de las pruebas con la solución de ingeniería de tráfico activada se ve que el promedio de paquetes perdido es de 7,9% . Esto corresponde al promedio que hace entre los 0% de paquetes perdidos cuando la solución TE es activada y por el camino corto solo viaja el tráfico sensible, y los primeros segundos donde existen gran cantidad de pérdidas ya que todavía no fue activada la solución y se encuentra compartiendo en enlace con el iperf TCP. De la *Tabla 5* se desprende que:

- Las pérdidas se redujeron a mas de un sexto
- El bandwidth aumentó un 80% aproximadamente

## 6 Conclusiones

### 6.1 Conclusiones

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

En cuanto al alcance del trabajo, se concluye que se llegó a cumplir el objetivo planteado. 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 de la red corporativa.

Una segunda conclusión a destacar es que se logró ir más allá del objetivo planteado al iniciar el trabajo. Si bien la idea principal era hacer implementar una solución TE basada en SDN, a esta se le agregó la creación del controlador SD-WAN con la interfaz gráfica para su administración. De esta forma se comprobó que resulta extremadamente útil para que administrador tener una interfaz que le permita realizar configuraciones abstrayéndose de los detalles de capas inferiores.

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, menores tasas 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 para alcanzar los requisitos deseados. Esto es claramente 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. Sin embargo, mediante una solución SDN, esto no solo posible, sino que facilita su implementación de manera considerable. Por otra parte, se vio que es posible hacer cambios en tiempo real sin que el servicio sufra interrupciones o degradaciones.

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 únicamente el ancho de banda disponible, también se podrían haber considerado otras métricas de calidad de servicio como delay o pérdida de paquetes).

En segundo lugar, se ve con buenos ojos aplicar un trabajo de frontend para implementar una interfaz gráfica aún mas amigable a la ya existente. 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 un menú interactivo fue para ayudar la labor del administrador y hacer mas rápido e intuitivo el trabajo, con lo cual mejorando la interfaz esto sería aun mas práctico.

Como un tercer punto pendiente queda la realización de pruebas de ingeniería de tráfico para IPv6. Si bien el escenario como el controlador SD-WAN permiten la configuración de IPv6 tal cual se hace con IPv4, mediante la utilización de iperf3 es posible realizar y corroborar su correcto funcionamiento. Para este trabajo si bien esto escapa del alcance, se ve como un punto interesante a futuro las pruebas con este protocolo de red.

Una cuarta 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, permitiendo filtrar tanto a nivel de enlace, red, transporte o aplicación. 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 a su vez sean más prioritarios que otros.

## Bibliografía

- [1] Logicalis, “*Software Defined Networks: el future de las arquitecturas de red*”. [En línea] Disponible <https://www.la.logicalis.com/globalassets/latin-america/logicalisnow/revista-20/lnow20-nota-42-45.pdf>
- [2] ditUPM, “*Software Defined WAN (SD-WAN)*”. Material de la asignatura SDNV de la Universidad Politécnica de Madrid
- [3] Nuage Networks, “*Virtualized Network Services*”. [En línea] Disponible <https://onestore.nokia.com/asset/183178>
- [4] Cisco, “*Cisco SD-WAN Solution Overview*”. [En línea] Disponible <https://www.cisco.com/c/en/us/solutions/collateral/enterprise-networks/sd-wan/nb-06-sd-wan-sol-overview-cte-en.html>
- [5] FlexiWAN, “*FlexiWAN documentation*”. [En línea] Disponible <https://docs.flexiwan.com/>
- [6] UCSD, “*B4: Expericnce with a Globally-Deployed Software Defined WAN*”. [En línea] Disponible <https://cseweb.ucsd.edu/~vahdat/papers/b4-sigcomm13.pdf>
- [7] Manav Bhatia, “*RoutingFreak!, The Classical Fish Problem in Routing*”. [En línea] Disponible <https://routingfreak.wordpress.com/2008/03/09/the-classical-fish-problem/>
- [8] Sdxcentral, “*What is Ryu Controller?*”. [En línea] Disponible <https://www.sdxcentral.com/networking/sdn/definitions/what-is-ryu-controller/>
- [9] ResearchGate, “*Controladores SDN, elementos para su selección y evaluación*”. [En línea] Disponible [https://www.researchgate.net/publication/320711755\\_Controladores\\_SDN\\_elementos\\_para\\_su\\_selección\\_y\\_evaluacion](https://www.researchgate.net/publication/320711755_Controladores_SDN_elementos_para_su_selección_y_evaluacion)
- [10] Gitgub, “*TrafficEngineering\_SDWAN*”. [En línea] Disponible [https://github.com/amuracciole/TrafficEngineering\\_SDWAN](https://github.com/amuracciole/TrafficEngineering_SDWAN)
- [11] Ryu, “*ryu.app.ofctl\_rest*”. [En línea] Disponible [https://ryu.readthedocs.io/en/latest/app/ofctl\\_rest.html](https://ryu.readthedocs.io/en/latest/app/ofctl_rest.html)
- [12] Aruba, “*SD-WAN Quality of service. Supplemental Guide*”



## **ANEXO A: ASPECTOS ÉTICOS, ECONÓMICOS, SOCIALES Y AMBIENTALES**

### **A.1 INTRODUCCIÓN**

Este proyecto se encuentra enmarcado en el desarrollo del Trabajo Fin de Master Universitario en Ingeniería en Redes y Servicios Telemáticos (MUIRST) dentro del marco del Departamento de Ingeniería de Sistemas Telemáticos (DIT) de la Escuela Técnica Superior de Ingenieros en Telecomunicaciones (ETSIT) de la Universidad Politécnica de Madrid (UPM).

Se tiene como objetivo la realización de una solución de ingeniería de tráfico que permita potenciar el tratamiento y encaminamiento de paquetes que viajan por una red SDN WAN formada por conmutadores. Se busca poder aplicar calidad de servicio a los diferentes tipos de tráficos de una manera mas flexible, rápida y sencilla de la que hoy existe con las redes tradicionales. A su vez busca una forma de realizar configuraciones de manera casi instantánea y en tiempo real a partir del estado actual de los enlaces.

Las redes definidas por software permiten realizar configuraciones mas específicas, de una manera más rápida, fiable y barata. A su vez abre la posibilidad a que una empresa pueda contratar los servicios a mas de un proveedor en simultaneo sin la necesidad de estar sujeto a uno solo como muchas veces ocurre actualmente. Esto posibilita una mayor competencia, volviéndose un mercado mas justo. Gracias a protocolos estandarizados y la permisividad que ofrece el software, es posible diseñar una solución conformada por equipos multivendor y mas de un proveedor de servicios.

Una implementación como la que se detalla en este trabajo presenta impactos económicos tanto para el operador de la red que debe realizar una fuerte inversión de equipamiento y capacitación del personal, como para el cliente final, al cual le permite potenciar aún mas sus aplicaciones de negocio. Sin embargo no se han identificado impactos sociales, geográficos, éticos o ambientales.

### **A.2 DESCRIPCIÓN DE IMPACTOS RELEVANTES RELACIONADOS CON EL PROYECTO**

A continuación se muestra una tabla la cual detalla los principales impactos identificados en este proyecto.

<b>Descripción</b>	<b>Grupos o sectores afectados e impactos</b>	<b>Implicaciones</b>
Gestión y configuración de soluciones SD-WAN.	<u>Grupos:</u> Técnicos e ingenieros de redes.  <u>Impacto positivo:</u> Búsqueda de personal calificado.	<u>Económica:</u> Inversión de capital humanos.
Mayores prestaciones sobre redes en comparación a las tradicionales.	<u>Grupos:</u> Operadores de red.  <u>Impacto positivo:</u> Nuevos modelos de negocios orientado a servicios.	<u>Económica:</u> Inversión en materiales y gestión de contabilidad.
Potenciar el negocio	<u>Grupos:</u> Clientes finales  <u>Impacto positivo:</u> Permite alcanzar mayores niveles de calidad en las aplicaciones de negocios	<u>Económica:</u> Ahorro de los servicios WAN empresariales contratados.

### A.3 ANÁLISIS DETALLADO DE ALGUNO DE LOS PRINCIPALES IMPACTOS

El punto mas destacado que provee este trabajo es relacionado al impacto económico positivo que traerá tanto a los operadores de red como a los usuarios finales. Hoy en día es muy costoso mantener un enlace MPLS para conectar dos oficinas mediante un túneles privados. Sin embargo con una solución de este tipo, donde se permita hacer un mejor uso de la infraestructura existente y un manejo mas rápido y eficiente de la red; hace que los costos de mantenimiento y configuración disminuyan, aumentando las ganancias a la vez que bajan los precios.

El perfil de ingeniero en telecomunicaciones será muy valorado con estas nuevas soluciones de redes definidas por software ya que son las personas que cuentan con conocimiento de programación y de diseño de redes.

### A.4 CONCLUSIONES

A modo de conclusión se determina que este trabajo no cuenta con impactos negativos considerables a nivel social, ético, económico ni medioambiental. Se considera que la huella ecológica que puede causar una solución de este tipo es mínima y de existir estaría relacionada al consumo eléctrico de los equipos de la red. Sin embargo, el impacto que tiene la virtualización en un escenario así hace que se precise de menos hardware para operar. Reduciendo así el consumo eléctrico.

Los impactos identificados en este proyecto benefician tanto a las empresas operadoras de red como a los usuarios finales. A su vez incentiva a la comunidad investigadora a desarrollar nuevas ideas de ingeniería de tráfico para que luego sean aplicadas y se obtengan mejores resultados de calidad de servicio en las redes WAN.



## ANEXO B: PRESUPUESTO ECONÓMICO

### COSTE DE MANO DE OBRA (coste directo)

Horas	Precio/hora	Total
450	15 €	6750 €

### COSTE DE RECURSOS MATERIALES (coste directo)

	Precio de compra	Uso en meses	Amortización (en años)	Total
Ordenador personal (Software incluido).....	1.500,00 €	6	5	150,00 €
Mobiliario de trabajo	400,00 €	6	5	40,00 €
Otro equipamiento				

### COSTE TOTAL DE RECURSOS MATERIALES

190,00 €

GASTOS GENERALES (costes indirectos)	15%	sobre CD	1.041,00 €
BENEFICIO INDUSTRIAL	6%	sobre CD+CI	478,86 €

### MATERIAL FUNGIBLE

Impresión	- €
Encuadernación	- €

SUBTOTAL PRESUPUESTO	8.459,86 €
IVA APPLICABLE	21% 1.776,57 €
TOTAL PRESUPUESTO	10.256,43 €