|  |
| --- |
| Universidad Politécnica de Madrid  Escuela Técnica Superior de Ingenieros de Telecomunicación |
| etsitupm2007-1343x385.gif |
| **DESARROLLO DE APLICACIONES DE INGENIERÍA DE TRÁFICO EN REDES WAN BASADAS EN SOFTWARE** |
| **TRABAJO FIN DE MÁSTER** |
| **Andrés Jorge Muracciole Vázquez** |
| 2020 |

|  |
| --- |
| Universidad Politécnica de Madrid  Escuela Técnica Superior de Ingenieros de Telecomunicación |
| **Máster Universitario en  Ingeniería de Redes y Servicios Telemáticos** |
| **TRABAJO FIN DE MÁSTER** |
| **desarrollo de aplicaciones de ingeniería de tráfico en redes wan basadas en software** |
| Autor  **Andrés Jorge Muracciole Vázquez** |
| Director  **Nombre y apellidos del director** |
| Departamento de Ingeniería de Sistemas Telemáticos  2020 |

# Resumen

Este documento forma parte de un trabajo de fin de master el cual trata acerca del estudio e implementación de una solución de ingeniería de tráfico para una red WAN basada en software. El objetivo de la misma es poder realizar una implementación que sea capaz de optimizar el camino que debe cursar el tráfico en una red WAN en función de las necesidades de las aplicaciones y el estado de la red en tiempo real.

En primer instancia se estudiarán los conceptos fundamentales de una red WAN tradicional identificando sus limitaciones que motivan su transformación hacia redes basadas en software y aplicaciones SDN )Software Define Network). Luego se mencionará cómo está conformada una red SD-WAN de forma de entender mejor este concepto y así poder adentrarse en la parte central del trabajo que consistirá en el diseño e implementación de una solución de tráfico para este tipo de redes. Aquí se detallarán no solo los conceptos relevantes de la misma sino que también se explicará el funcionamiento y desarrollo de la solución. Con ello se buscará no solo que el lector entienda el trabajo técnico realizado sino también ayudar a reflexionar sobre el potencial de este tipo de soluciones en el mercado actual. Finalmente se realizarán una serie de pruebas prácticas donde se reflejará el funcionamiento de la solución establecida y se verá el camino que toma el tráfico en función de las prestaciones de la red. Por último se mencionarán a una serie de ejemplos prácticos donde se puede ver beneficiado el negocio con una solución de este tipo, finalizando con una serie de conclusiones a modo de resumen general.

# Abstract

# Índice general

[Resumen i](#_Toc42539972)

[Abstract iii](#_Toc42539973)

[Índice general v](#_Toc42539974)

[Índice de figuras vii](#_Toc42539975)

[Siglas ix](#_Toc42539976)

[1 Introducción 1](#_Toc42539977)

[2 Estado del arte 2](#_Toc42539978)

[2.1 Soluciones comerciales y académicas 3](#_Toc42539979)

[2.1.1 Nuage 3](#_Toc42539980)

[2.1.2 Viptela 5](#_Toc42539981)

[2.1.3 FlexiWAN 6](#_Toc42539982)

[2.2 Otros trabajos de investigación con ingeniería de tráfico SDN 7](#_Toc42539983)

[2.2.1 B4 8](#_Toc42539984)

[3 Definición del caso de uso 9](#_Toc42539985)

[4 Aplicación de ingeniería de tráfico en SD-WAN 10](#_Toc42539986)

[4.1 Pruebas preliminares sobre servicios SD-WAN de código abierto 10](#_Toc42539987)

[5 Resultados obtenidos 12](#_Toc42539988)

[6 Conclusiones y líneas futuras 13](#_Toc42539989)

[Bibliografía 15](#_Toc42539990)

# Índice de figuras

[Figura 1 – Red sin interrupciones SD-WAN 4](#_Toc42540006)

[Figura 2 – Arquitectura Nuage 5](#_Toc42540007)

[Figura 3 – Arquitectura Viptela 6](#_Toc42540008)

[Figura 4 – Diagrama de bloques de alto nivel de la arquitectura flexiWAN 7](#_Toc42540009)

[Figura 5 – Visualización el flexiWAN-router desde la interfaz central manage 10](#_Toc42540010)

[Figura 6 – Estadísticas de bps y pps en un flexiWAN-router 11](#_Toc42540011)

# Siglas

SDN: Software Define Network

SD-WAN: Software Define Wide Area Network

CPD: Centro de Procesamiento de Datos

ISP: Internet Service Provider

WAN: Wide Area Network

LAN: Local Area Network

QoS: Quality of Service

# Introducción

-Ps de contexto y motivación. Falta de flexibilidad de las redes WAN tradicionales. La migración de las aplicaciones de negocio a la nube…. El aumento de la demanda de los servicios SD-WAN.

-Ps las redes están sufriendo cambios mediante la incorporación de tecnologías como SDN, NFV. Los servicios WAN se están adaptando…

\_Ps Falta de flexibilidad para aplicar en tiempo soluciones de ingenieria de tráfico….

-Ps en este TFM se ha desarrollado una app para….

-Ps herramientas.

-Ps se han obtenido que verifican el..

# Estado del arte

Las redes LAN (Local Area Network) son sin duda las mas utilizadas a nivel empresarial. Allí radican los servicios internos de una empresa así como también los activos tales como impresoras, registros y servidores de información, los cuales en un principio solo interesa que se tenga acceso a ellos 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. Estas redes 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 estas, 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 y es por esto que es imposible asegurar y estimar la calidad de servicio a la que están expuestos. En definitiva no hay que olvidarse que internet es *Best Effort*, con lo cual no hay forma de priorizar los paquetes. A pesar de que por algún motivo se modifiquen las cabeceras, una vez que entra a la red de 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 de forma de asegurarse un nivel de QoS aceptable para 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 que tipo de tráfico es mas sensible y en función de ello enviarlo por el canal con mayores prestaciones. A pesar de ello, estos servicios suelen ser bastante costosos para las empresas y poco flexibles para el proveedor.

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

Es a raíz de todo esto que las redes SDN vienen a simplificar estos asuntos dando mas flexibilidad de una manera mas ágil y económica. Con este nuevo paradigma un operador puede al cabo de unos pocos minutos levantar un nuevo sitio remoto, realizar configuraciones y establecer criterios de flujo para decidir bajo demanda el camino mas óptimo por el cual enviar los paquetes en función de las prestaciones que se requieran. A su vez, con la inclusión de la virtualización de funciones de red (VNF) es posible realizar un despliegue de equipamiento de red virtualizado como si fuera físico disminuyendo el tiempo de ejecución y puesta en marcha significativamente respecto al método tradicional. Por otra parte, muchas veces los enlaces WAN son contratados a diferentes empresas; sin embargo, las soluciones SD-WAN son independientes a esto, permitiendo trabajar con ellas sin importar si el proveedor es o no el mismo que brinda la solución SD-WAN. En definitiva permite unificar las redes WAN y tratarla como si fuera “una sola”, bajo demanda y realizando balanceos de carga de ser necesario.

## Soluciones comerciales y académicas

Sin duda alguna, con la incursión del las nuevas tecnologías y el cambio de paradigma respecto a las redes de comunicación tradicionales con SDN y SD-WAN, abren el abanico de soluciones de ingeniería de tráfico y es allí donde varios proveedores comenzaron a ver la importancia de las mismas y el potencial que podrían llegar a alcanzar en caso de migrar su red hacia estas. Es por ello que existen diversas soluciones, con la desventaja de que son en su totalidad servicios de pago con lo cual no es posible realizar pruebas de laboratorio sin tener que pagar por ello. Algunas implementaciones cuentan con licencias trial pero están muy limitadas en cuanto a su uso y no permiten realizar lo que se espera en este trabajo. A continuación se procederá a describir tres soluciones comerciales de servicio SD-WAN para entender un poco mejor de que tratan, haciendo mayor hincapié en la solución de *Flexiwan* ya que si bien es solución paga, esta es de código abierto, por lo que fue posible también realizar una serie de pruebas que se detallarán en el punto 4.1 de este documento.

### Nuage

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

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

Por otra parte esta Nuage permite ocultar la complejidad de la red WAN empresarial estableciendo una WAN extremo a extremo que posibilita conectar centros de datos privados, sucursales y servicios de nube pública o privada como se puede ver en la Figura 1.

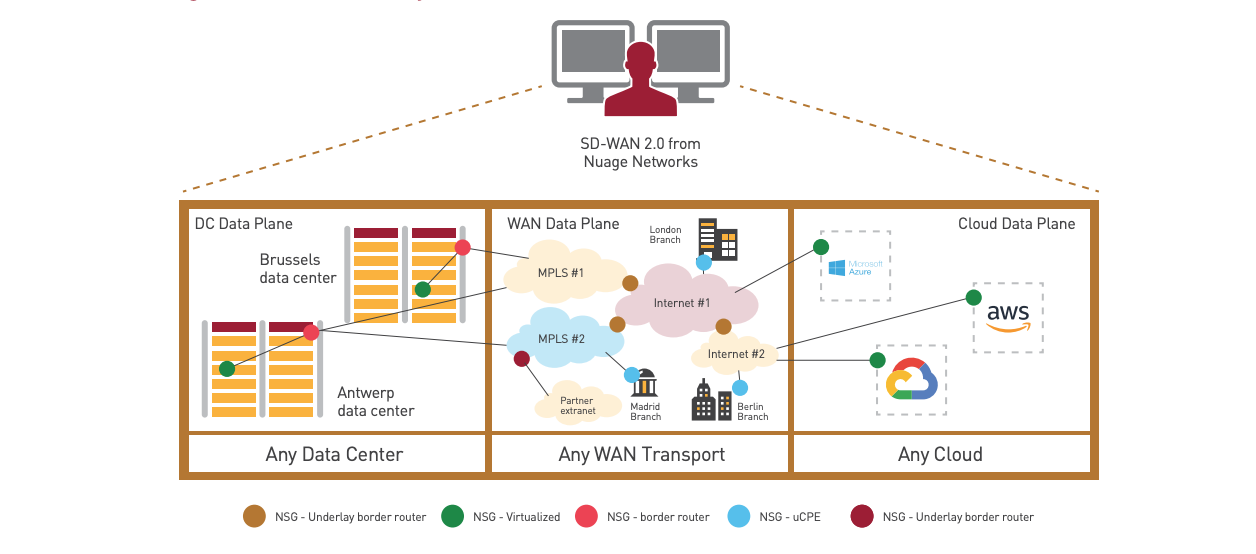


Figura 1 – Red sin interrupciones SD-WAN

La Figura 2 muestra la arquitectura de Nuage. El VSD (Virtualized Services Directory) permite al administrados SD-WAN definir y aplicar políticas a la red de una manera fácil y de forma gestionada y centralizada. A su vez implementa la funcionalidad de recolección de datos de forma periódica para la creación de informes y reportes de la red, así como también alertas en caso de que un tráfico haya sobrepasado un umbral determinado. Todas estas funciones se pueden desplegar en un portal personalizable con widgets. Las funciones de red se pueden seleccionar mediante el catálogo de VSD, pudiendo optar entra VNF´s de firewalls, IPSec, NAT, balanceo de carga y gestión de dominios, entre otros.

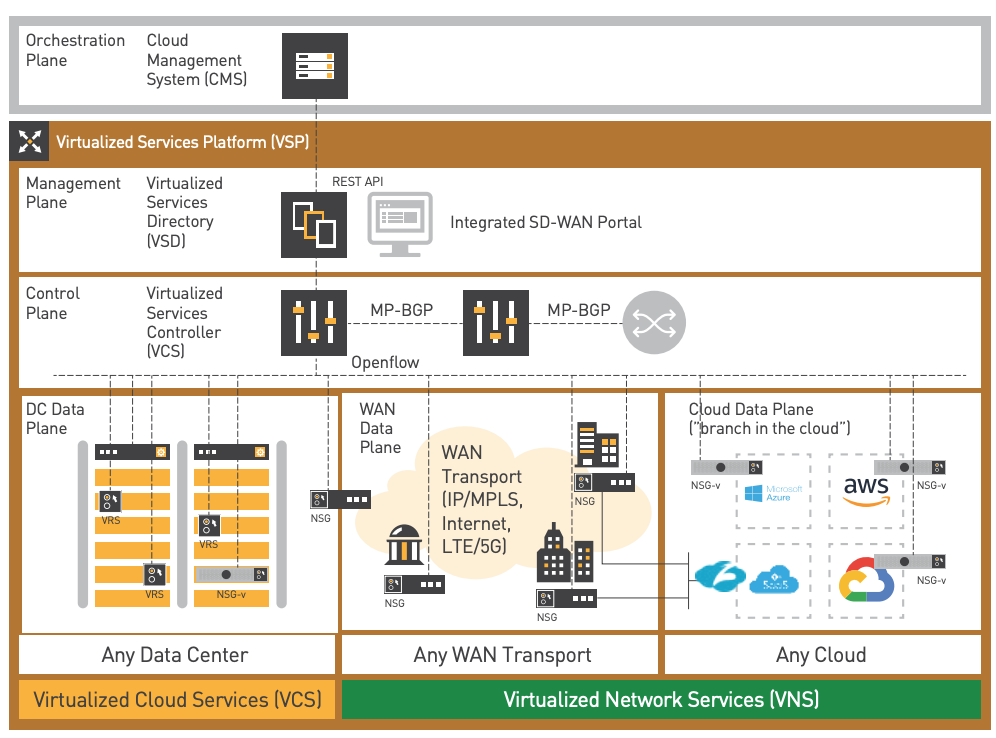


Figura 2 – Arquitectura Nuage

### Viptela

**Viptela** [i[[1]](#endnote-1)] es la solución SD-WAN adquirida por Cisco que permite abstraerse del medio de transporte y tener visibilidad en tiempo real del estado de la red para así poder por ejemplo tomar decisiones de enrutamiento. Viptela está conformada por cuatro componentes, cada uno asociado a un plano diferente tal cual se ve en la Figura 3. Por un lado se encuentra el vEdge que representa el router en las oficinas. Este es quien se encarga de establecer comunicaciones seguras con los demás vEdges (virtualizados o físicos) mediante túneles IPSec y son quienes ejecutan las políticas implementadas por el vSmart.

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

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

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

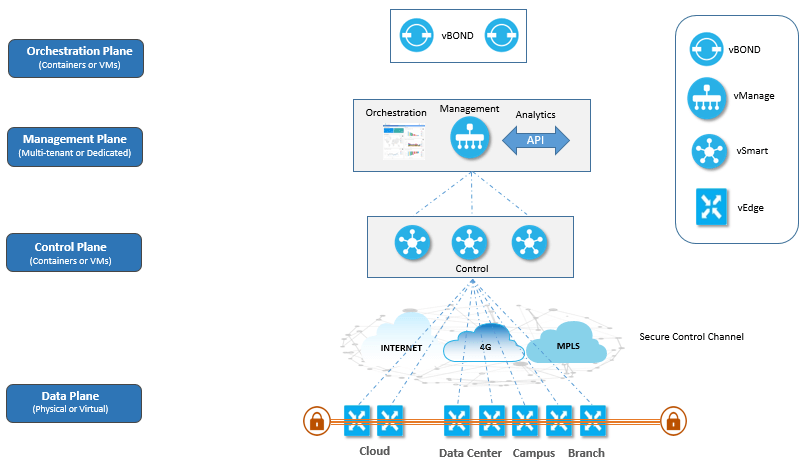


Figura 3 – Arquitectura Viptela

### FlexiWAN

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

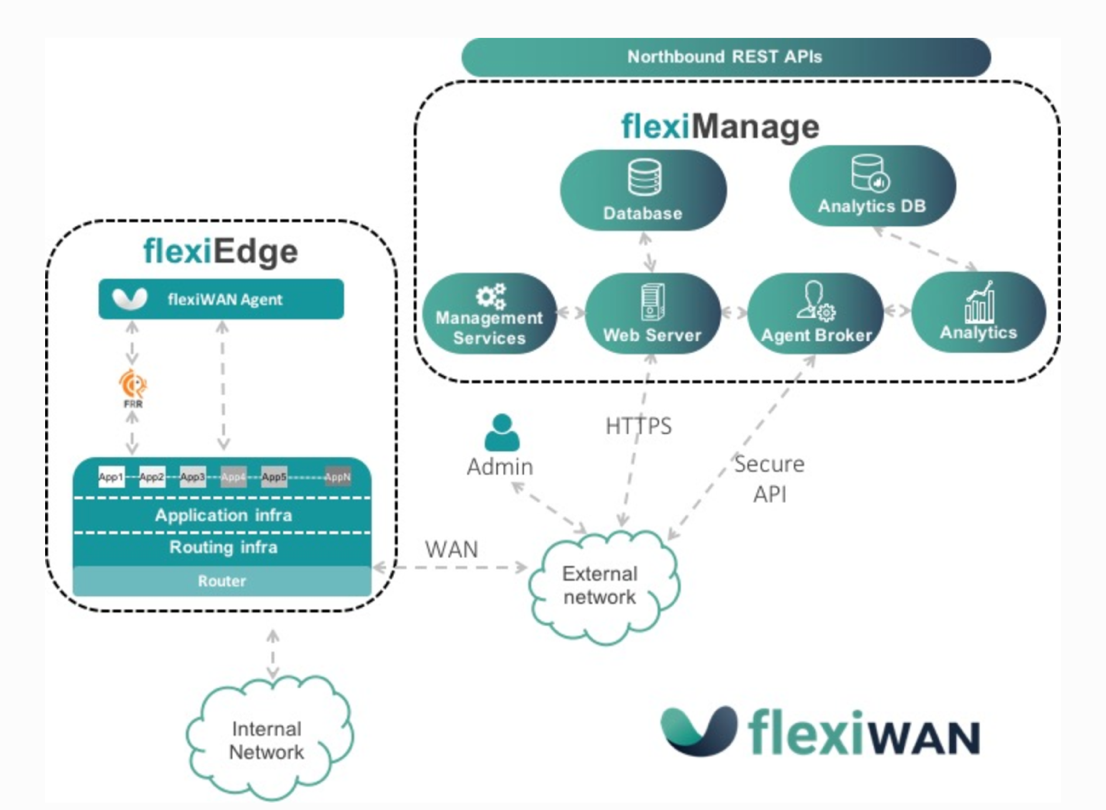


Figura 4 – Diagrama de bloques de alto nivel de la arquitectura flexiWAN

## Otros trabajos de investigación con ingeniería de tráfico SDN

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

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

* *Minimizar la congestión de red*: Mediante técnicas de denegación de acceso es posible impedir el acceso a los recursos congestionados, redistribuyendo los flujos mediante rutas multicaminos.
* *Reducir el retardo extremo a extremo*: Técnicas como CSFP (*Constrained Shortest Path First*) permite determinar el camino lógico mas corto entre dos puntos de la red, minimizando así el retardo.
* *Minimizar la pérdida de paquetes:* Esto se puede dar por diversos motivos como caída o saturación de los enlaces. Para disminuir este factor se suelen emplear caminos de respaldo o protocolos de convergencia para disminuir al máximo este problema.
* *Priorización de tráfico:* Es posible realizar técnicas para determinar diferentes niveles de tráfico y así tratarlos de forma diferenciada. Esto hace que se envíe un tipo de paquete mas sensible antes que otro lo cual puede traer como consecuencia una mejoría en la calidad de experiencia de dicho servicio. El caso mas sencillo de visualizar es para el tráfico de streaming donde se requiere que este sea lo mas fluido posible.

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

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

### B4

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

El algoritmo permite determinar los túneles virtuales que se deben programar en la red para que se cumplan con los requisitos de las aplicaciones, así como también el ancho de banda asignado a cada una. Este es asignado en función de las prioridades de las aplicaciones las cuales vienen dadas a partir del volumen y la sensibilidad frente a retardos

# Definición del caso de uso

Descripción del escenario “real”. Descripción, dos sedes remotas de una corporación, o si estamos CPDs, dibujo explicativo. Diagrama que represente ese caso de uso. Topología de pez

# Aplicación de ingeniería de tráfico en SD-WAN

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

En problema a abordar consiste en el desarrollo e implementación de una solución de ingeniería de tráfico SD-WAN que sea capaz de poder seleccionar bajo demanda el camino que debe tomar un tipo de tráfico determinado a partir de dos componentes esenciales: estado de la red en tiempo real y parámetros de QoS. El escenario a tratar (sección 4.3) busca parecerse lo mas posible a la realidad (ver 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 cual de ellos es el más óptimo para el tipo de tráfico a enviar. Se entiende por “tráfico sensible” todo aquel que previamente haya definido el administrador de red como tal y este se especifica en la sección 4.

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

Una vez estudiada la teoría de flexiWAN en el apartado 2.1.3, se procedió a implementar un escenario para ver en funcionamiento de las prestaciones que ofrece la plataforma en la versión gratuita. En primer lugar se crearon y configuraron en VirtualBox dos máquinas virtuales Ubuntu con conexión a internet para que tengan conectividad con el flexiManage las cuales simulan ser dos sitios remotos. En ambas fue necesario asociarles un token de autenticación (creado en el flexiManage) para reconocerlos como sitios seguros y establecer una canal de comunicación contra ellos. Una vez hecha la autenticación, los routers virtuales aparecen en el dashboard dentro de la sección Inventory/Devices and manage como se aprecia en la Figura 5.

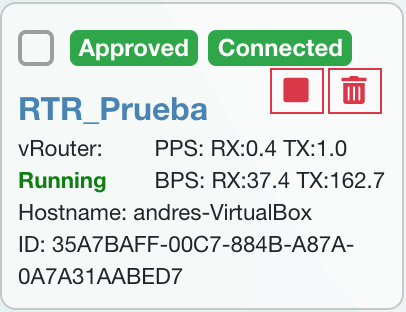


Figura 5 – Visualización el flexiWAN-router desde la interfaz central manage

Teniendo los sitios remotos configurados y activos en el flexiManage es que se puede comenzar a realizar las visualizaciones y configuraciones que se deseen como por ejemplo cambiar de forma remota las direcciones IP públicas y privadas del router, así como también crear nuevas interfaces. Todo esto se hace de forma centralizada y en pocos minutos lo cual proporciona una flexibilidad y rapidez en la solución que no se obtendría con una solución tradicional. Por otro lado se probó crear túneles virtuales entre los flexiEdges de forma de darle conectividad entre ellos. Una vez hecho esto la herramienta permite medir la latencia y los paquetes perdidos en dicho canal. Esto se hace también de forma centralizada desde la sección Inventory/Tunneles, lo cual muestra otra vez una gran ventaja respecto a sistemas tradicionales donde es necesario realizar configuraciones en ambos extremos para establecer un túnel.

En cuanto a las visualizaciones que permite la plataforma, estas son bastante limitadas pero aceptables. Por un lado es posible obtener las tablas de ruteo de cada flexiEdge y modificarlas o agregar nuevas rutas de ser necesario. También admite obtener estadísticas en tiempo real del tráfico cursado tanto en BPS (bits por segundo) o PPS (paquetes por segundo) como se muestra en la Figura 6. Todas las indicaciones fueron tomadas del manual de flexiWAN [iii].

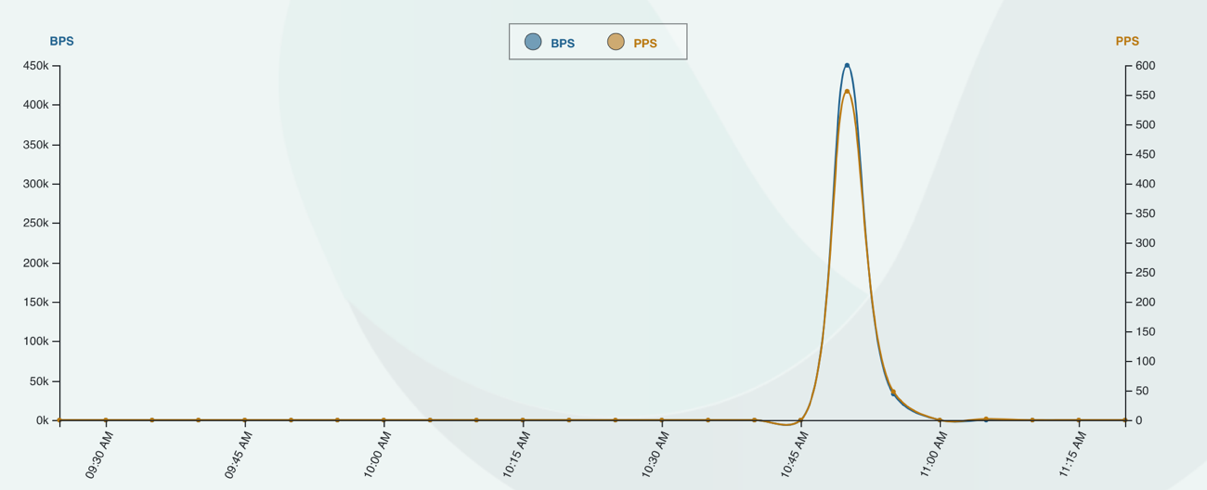


Figura 6 – Estadísticas de bps y pps en un flexiWAN-router

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.

## Controlador Ryu

SDN cuenta con una arquitectura muy diferente a la tradicional ya que se busca desacoplar de forma lógica y en algunos casos física el plano de control [[2]](#footnote-1) del plano de datos [[3]](#footnote-2) haciendo así la red mas programable, automatizada y controlable. Con estas soluciones nace el concepto de controlador, el cual oficia como “cerebro” de la red, teniendo una visión general de la red y comunicándose con los equipos mediante protocolos estandarizados como 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 7 [v] se puede apreciar una visión lógica de este tipo de arquitecturas SDN donde queda en evidencia el desacople del plano de datos (capa de infraestructura) y el plano de control (capa de control). En este último se encuentra el controlador SDN el cual se comunica hacia el norte (capa de aplicaciones) mediante API´s haciendo así muy gestionable y programable en función del uso requerido, así como también permitiendo la integración con aplicaciones de negocio, calidad de servicio, entre otras. Por otro lado, la comunicación hacia los dispositivos de red, comúnmente denominados conmutadores, se hace como se mencionó anteriormente por medio de protocolos estándares como por ejemplo OpenFlow mediante paquetes *packet-in* [[4]](#footnote-3) y *packet-out* [[5]](#footnote-4). Mediante estos, el controlador se comunica con los routers y switches 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.

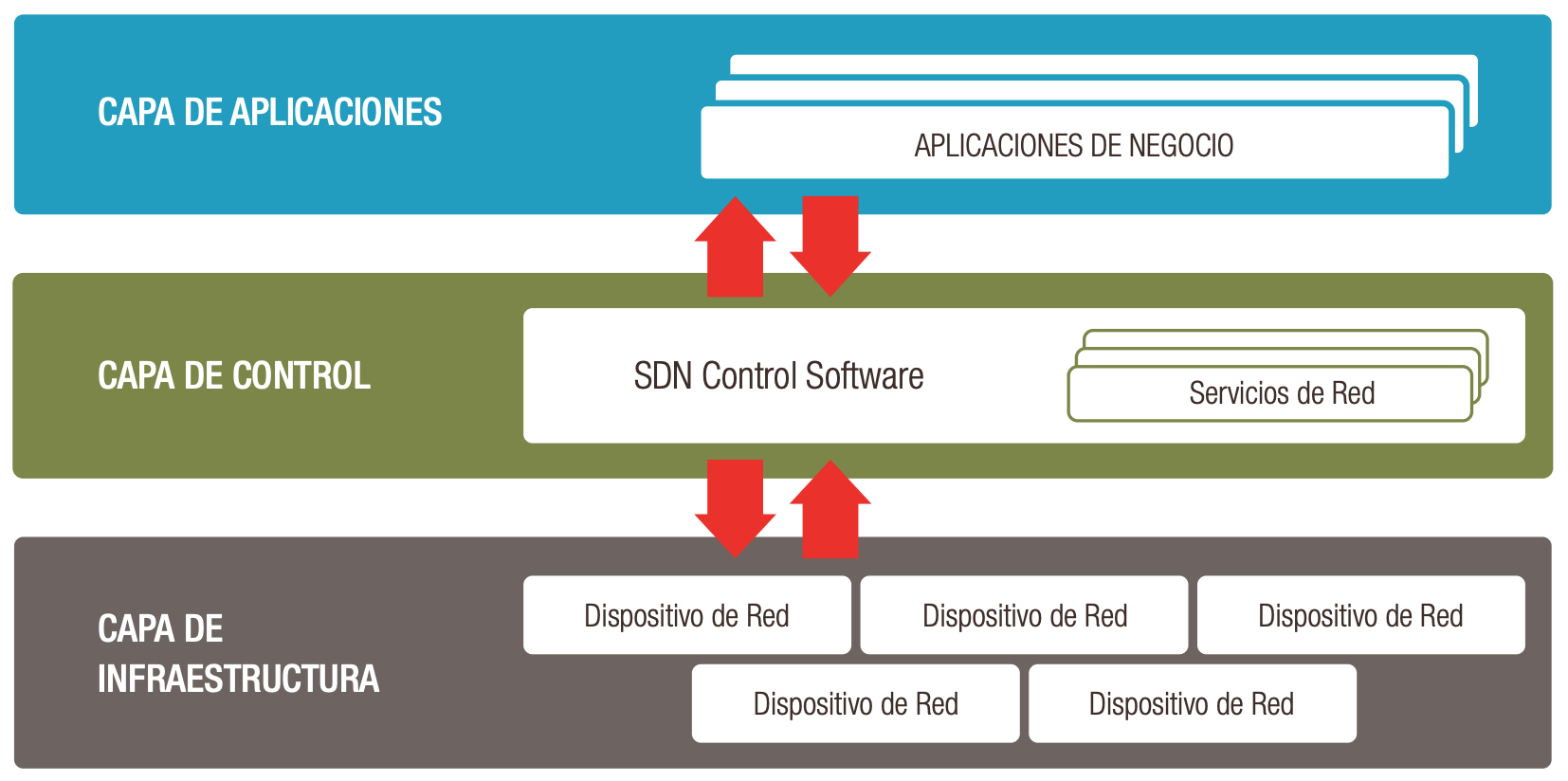


Figura 7 – Arquitectura SDN

El controlador elegido para esta implementación es Ryu [[6]](#footnote-5) debido a una serie de factores que lo diferencian del resto. En primer lugar es una solución Open Source SDN diseñada para incrementar la agilidad de la red facilitando la administración y adaptación de cómo se maneja el tráfico [vi]. Cuenta con una gran cantidad de aplicaciones creadas por desarrolladores y publicadas en diversos portales abiertos y foros reconocidos. De esta forma se potencializa el uso de la plataforma ya que se genera un clima acorde para la transferencia de conocimientos y potenciamiento de la herramienta aumentando así el alcance de la misma. En segundo lugar soporta NETCONF, extensiones Nicira y diversas versiones de OF (OpenFlow) [vi], en especial la 1.3 que es la utilizada en este trabajo. Provee una API REST básica pero a la vez poderosa que ayuda a diseñar soluciones acordes para cada desarrollador, a parte de una interfaz web muy limitada pero que al ser de código abierto permite su adaptación. Este framework está escrito en lenguaje Python y apoyado sobre plataforma Linux lo cual permite la virtualización con Mininet y OpenvSwitch. Esta característica permite a los administradores crear redes virtuales de forma dinámica, disociadas de las redes físicas para por ejemplo satisfacer la ampliación de la capacidad sin afectar los flujos existentes [vii].

Otro motivo por el cual se eligió este controlador es que permite configurar flujos de forma proactiva y reactiva. La manera proactiva consiste en configurar la regla de flujo antes de que el paquete llegue al switch OpenFlow. De esta forma una vez que un paquete lleue al mismo, este ya conoce el tratamiento que debe realizar, reduciendo así el tiempo de encaminamiento. Esto se puede hacer mediante la API REST propia de Ryu. Por otro lado, una configuración reactiva se produce cuando el conmutador recibe un paquete que no coincide con ninguna de sus entradas en la tabla de flujo y debe realizarle una consulta mediante packet-in al controlador, respondiéndole este con un packet-out la regla de como tratarlo. Esto a gran escala genera un tráfico considerable en la red así como también un retardo (tiempo de paquetes del controlador al conmutador y viceversa, tiempos de procesamiento de los paquetes y tiempo de llenado de tablas de flujo) si se lo compara con el método proactivo [vii].

El código del controlador Ryu se encuentra configurado el Python y está alojado el Github [[7]](#footnote-6). La comunidad Ryu es quien se encarga de efectuarle mejoras, solucionando los posibles fallos y agregando nuevas funcionalidades.

### Traffic Monitor

Conjuntamente con el controlador Ryu se ejecuta el script de Traffic Monitor [[8]](#footnote-7) el cual permite obtener en tiempo real el estado de los enlaces de la red de una forma rápida y sencilla. Cabe destacar que la versión utilizada en este trabajo es una extensión modificada del controlador de código abierto creada por la comunidad de programadores, la cual permite visualizar a nivel de consola los links de los conmutadores, el estado de los mismos, el tráfico instantáneo cursado por las interfaces y un histórico de la cantidad de paquetes y bytes enviados, recibidos y con errores. Por otro lado, existe una instancia que cuando es levantado por primera vez, realiza un escaneo rápido de la red y obtiene la topología de red, viendo así las conexiones entre conmutadores e imprimiéndolas en pantalla tal cual se ve en la sección *Link Port* de figura 8. A modo de ejemplo se puede ver que el conmutador con datapath *2560* (CONM\_A) se encuentra conectado por el puerto 1 al puerto 1 del *3072* (CONM\_C) y por el puerto 2 al 1 del *2816* (CONM\_B). De forma análoga de puede ver el resto de las conexiones. Esta nomenclatura tan particular la obtiene a partir de convertir en decimal la dirección *hwaddr.* A modo de ejemplo para el conmutador A, el valor 0x000000000A00 equivale a 2560 en base decimal.

*<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 1 – Extracto de configuración de CONM\_A

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

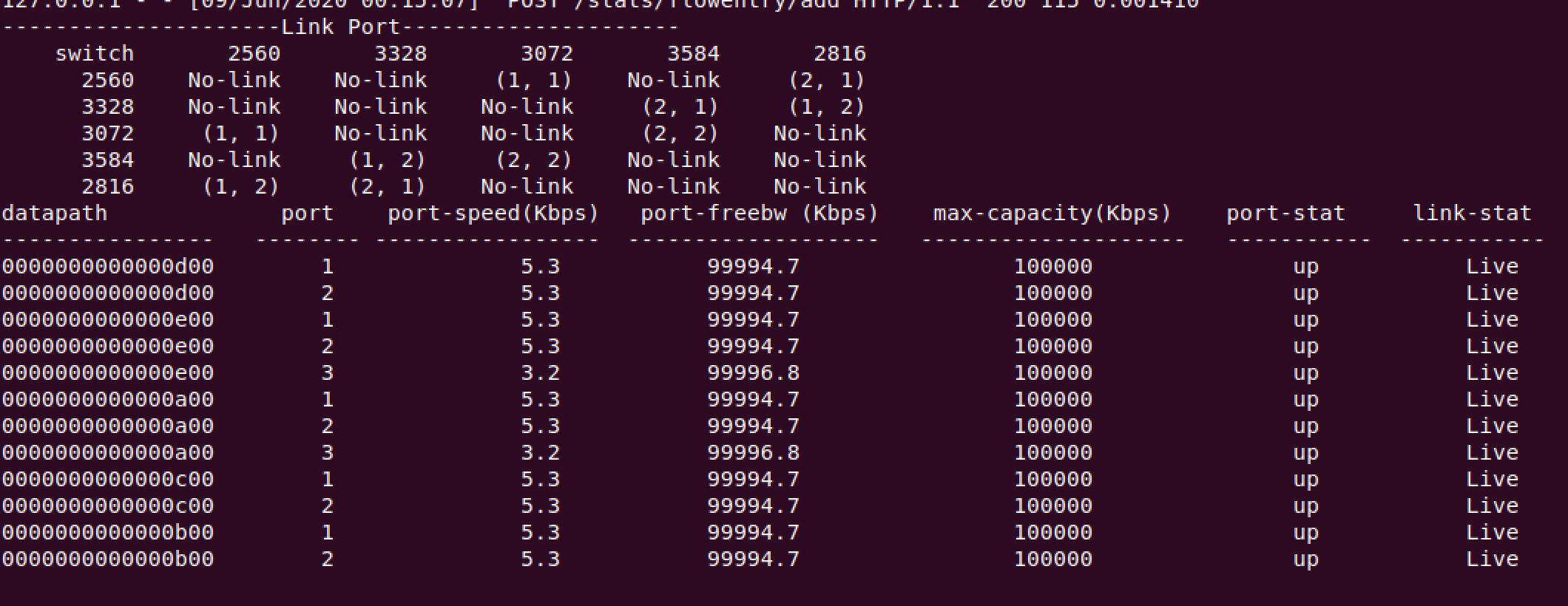


Figura. 8 – Tablas de topología y puertos por tráfico del programa Traffic Monitor de Ryu

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

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 8 a un fichero csv llamado “*Table.csv*”. Esto permitirá hacer un tratamiento con estos datos según se detalla en la sección 4.4.

### Ofctl\_rest

Mediante la interfaz northbound el conmutador Ryu es capaz de interactuar con la API ofctl\_rest [viii]. Esta es esencial para el desarrollo de la solución en cuestión ya que mediante el método POST permite agregar, modificar o quitar entradas en las tablas de flujo de los conmutadores. La secuencia de la misma se detalla a continuación y se puede acompañar con la figura 9:

1. El usuario solicita por ejemplo agregar una entrada de flujo en un conmutador. Para ello establece una conexión TCP hacia el puerto 8080 donde se encuentra escuchando la API. Estos intercambian una serie de paquetes de sincronización TCP para luego realiza el POST HTTP. Por último se cierra la conexión.
2. En este momento el controlador realiza la función de pasarela, enviando mediante un paquete OpenFlow de tipo FLOW\_MOD [[9]](#footnote-8) la acción hacia el puerto TCP que tiene conectado hacia el controlador (este se encuentra escuchando en el 6633).
3. El conmutador modifica su tabla de flujo con la entrada configurada en el punto 1.

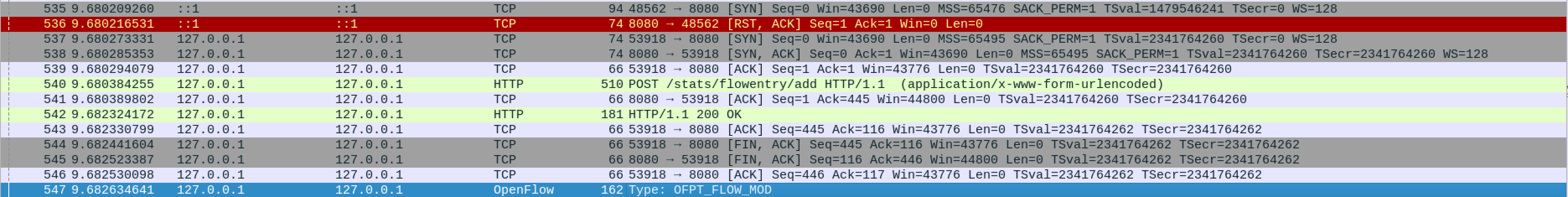


Figura 9 – Captura de tráfico ante solicitud de modificación de entrada de flujo hecha con Wireshark

## Escenario de red virtual

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

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

Estas dos premisas son importantes para poder luego poder implementar una ingeniería de tráfico que permita decidir en tiempo real el camino que deben tomar los paquetes. En cuanto a la “topología de pez” esta se puede apreciar en la figura 10 con los conmutadores A,B,C,D y E. Aquí se puede ver que para ir de R1\_edge a R2\_edge por la red de conmutadores es posible hacerlo por dos caminos diferentes. Uno pasando por CONM\_A, CONM\_C y CONM\_E, al cual llamaremos de aquí en más “short path” y otro conformado por CONM\_A, CONM\_B, CONM\_D y CONM\_E al cual se lo llamará “long path”. Lo interesante aquí no radica en la existencia de multicaminos sino en que uno de ellos es mas largo que el otro si se toma como medida la cantidad de saltos. Este problema es bien conocido y deja al descubierto un punto débil de los protocolos de ruteo tradicional como RIP u OSFPF ya que convierte un escenario multicamino como si fuera de uno solo.

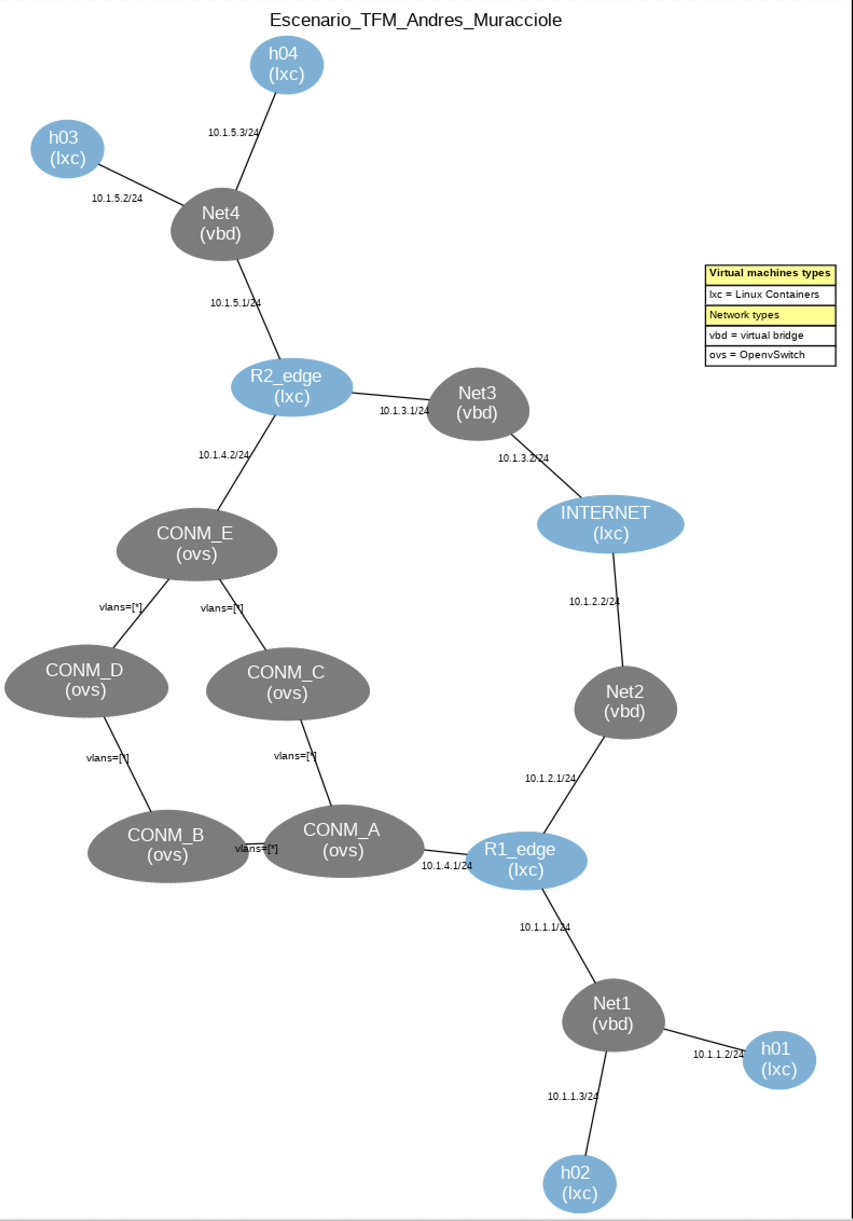


Figura 10 – Escenario virtual con red de conmutadores en topología de pez

En un escenario con ruteo clásico y suponiendo, como es el caso, que los enlaces son todos iguales, el encaminamiento se dará por el camino mas corto (como se puede ver de forma ilustrativa en la figura 11) ya que tiene menor métrica, inutilizando el camino mas largo. De esta forma se puede llegar a saturar la red rápidamente a pesar de tener un camino por el cual no se trafican datos [ix].

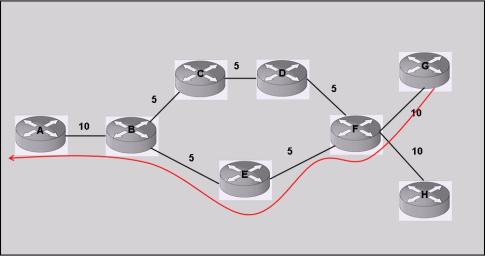


Figura 11 – Topología de pez

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 puede apreciar en la figura 12. Sin embargo esta solución es muy mala ya que lo que hace es migrar todo el tráfico de un camino al otro dejando ahora el camino corto sin utilización y volviendo a tener congestión en la red.

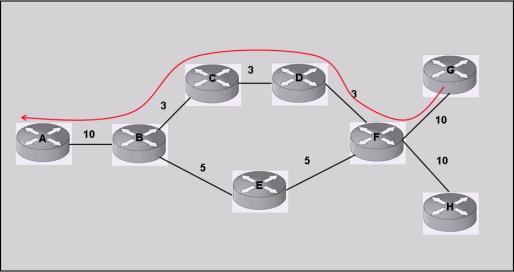


Figura 12 – Topología de pez

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

La figura 10 muestra el escenario realizado en VNX el cual está conformado por:

* 4 contenedores LXC que ofician de sistemas finales: h01, h02, h03 y h04
* 5 conmutadores OVS: CONM\_A, CONM\_B, CONM\_C, CONM\_D y CONM\_E
* 2 contenedores LXC-Vyos que ofician de routers: R1\_edge y R2\_edge
* 4 redes: Net1, Net2, Net3 y Net4
* 1 contenedor LXC que representa el enlace a internet: INTERNET

Cada interfaz tiene configurada una dirección IPv4 como se muestra en la tabla 1.

|  |  |  |  |
| --- | --- | --- | --- |
| NOMBRE | INTERFAZ | MAC | IPv4 |
| h01 | 1 | 00:00:00:00:01:01 | 10.1.1.2 |
| h02 | 1 | 00:00:00:00:02:01 | 10.1.1.3 |
| h03 | 1 | 00:00:00:00:03:01 | 10.1.5.2 |
| h04 | 1 | 00:00:00:00:04:01 | 10.1.5.3 |
| R1\_edge | 1 | 00:00:00:00:06:01 | 10.1.1.1 |
| R1\_edge | 2 | 00:00:00:00:06:02 | 10.1.2.1 |
| R1\_edge | 3 | 00:00:00:00:06:03 | 10.1.4.1 |
| R2\_edge | 1 | 00:00:00:00:07:01 | 10.1.5.1 |
| R2\_edge | 2 | 00:00:00:00:07:02 | 10.1.3.1 |
| R2\_edge | 3 | 00:00:00:00:07:03 | 10.1.4.2 |
| INTERNET | 1 | 00:00:00:00:05:01 | 10.1.2.2 |
| INTERNET | 2 | 00:00:00:00:05:02 | 10.1.3.2 |
| CONM\_A | - | 00:00:00:00:0A:00 (hwaddr) | - |
| CONM\_B | - | 00:00:00:00:0B:00 (hwaddr) | - |
| CONM\_C | - | 00:00:00:00:0C:00 (hwaddr) | - |
| CONM\_D | - | 00:00:00:00:0D:00 (hwaddr) | - |
| CONM\_E | - | 00:00:00:00:0E:00 (hwaddr) | - |

Tabla 1 – Direccionamiento L2 y L3 del escenario

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

### Wondershaper

## Desarrollo de aplicaciones SDN

4.1 Escenario de red con VNX

Figura VNX –show-map y la describes. Describes el xml

He desarrollado un escenario con VNX. Las sedes remotas se modelan como varios sistemas finales que se implementan como contenedores LXC.

Explica que has configurado el escenario con ipv6. Explica que utilizas un vyos. Cuenta que tienes un Ryu corriendo. Traffic monitoring – contar algo de ella y explicar por qué la has modificado y cuál es la modificación. Simple\_switch. Ofctl.rest

4.2 Desarrollo aplicaciones SDN

Explicar diagrama de relación entre las piezas de código que has programado. Puedes comentar en líneas generales que hace cada programa. Luego hacemos un zoom in sobre las piezas de código más importantes.

Explica para qué sirve cada pieza de código y explica el código (o parte) si es pertinente. Lo más importante: controlador SDWAN (main). Controlador SD-WAN permite implementar la solución de ingeniería de tráfico haciendo llamadas a las cajas (X,YZ) que se utilizan para programar… y haciendo llamadas a la aplicación traffic monitoring… explica que la lista de opciones (captura)

Opción 1: explícala y comenta cuales son las piezas de código que utiliza. (X,O,P). La X sirve para, la O hace tal cosa…

Opción 2: explícala y comenta cuales son las piezas de código que utiliza. (X,O,P)

Opciones 3: explícala y comenta cuales son las piezas de código que utiliza. (balanceo de carga,O,P) , y que expliques los parámetros que se utilizan para tomar esas decisiones de conmutación. Cómo se utiliza la información del traffic monitoring (+utilización)

1

2

3

4

# imprimo…

En la línea número 4 del código X lo que hago…

### Trñafico Sensible

### Monitor.py

# Resultados obtenidos

5.1 Experimento 1:

Diseño de experimento. Definición de los parámetros: asumo 70% utilización, tráfico iperf udp entre cliente h1 y cliente h4….

Resultados: pintas tu gráfica. Muestras las tablas openflow, los datos del iperf, ….

Los resultados de monitorización. Con una utilización del 50% estas son las tablas y este es el camino. Sin embargo, cuando la utilización es mayor al 70% estas son las tablas y este es el camino y estas son las gráficas.

5.2 Experimento 2:

# Conclusiones y líneas futuras

Conclusiones que son sobre el servicio y el uso de las tecnologías SDN, NFV… para aplicar soluciones de ingeniería de tráfico en tiempo real…

Conclusiones de los resultados.

Línea futura: programar una interfaz del controlador SD-WAN …

# Bibliografía

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

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

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

   **[iv]** Logicalis, Software Defined Networks: el future de las arquitecturas de red. [Online] de https://cseweb.ucsd.edu/~vahdat/papers/b4-sigcomm13.pdf

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

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

   **[vii]** ResearchGate, Controladores SDN, elementos para su selección y evaluación. [Online] de https://www.researchgate.net/publication/320711755\_Controladores\_SDN\_elementos\_para\_su\_seleccion\_y\_evaluacion

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

   **[ix]** RoutingFreak!, The Classical Fish Problem in Routing. [Online] de https://routingfreak.wordpress.com/2008/03/09/the-classical-fish-problem/ [↑](#endnote-ref-1)
2. Orientado a tratar el tráfico orientado a la gestión, mantenimiento y modificación de la red y sus componentes [↑](#footnote-ref-1)
3. Orientado a tratar el tráfico destinado a los servicios [↑](#footnote-ref-2)
4. Paquetes del conmutador al controlador [↑](#footnote-ref-3)
5. Paquetes del controlador al conmutador [↑](#footnote-ref-4)
6. https://ryu-sdn.org/index.html [↑](#footnote-ref-5)
7. https://github.com/faucetsdn/ryu [↑](#footnote-ref-6)
8. https://github.com/muzixing/ryu [↑](#footnote-ref-7)
9. Tipo de paquete OpenFlow que permite modificar el estado de un switch OpenFlow [↑](#footnote-ref-8)