MONITORIZACIÓN DE TRÁFICO EN REDES WAN DEFINIDAS POR SOFTWARE MEDIANTE CONTROLADOR RYU

Memoria del proyecto

Integrantes:

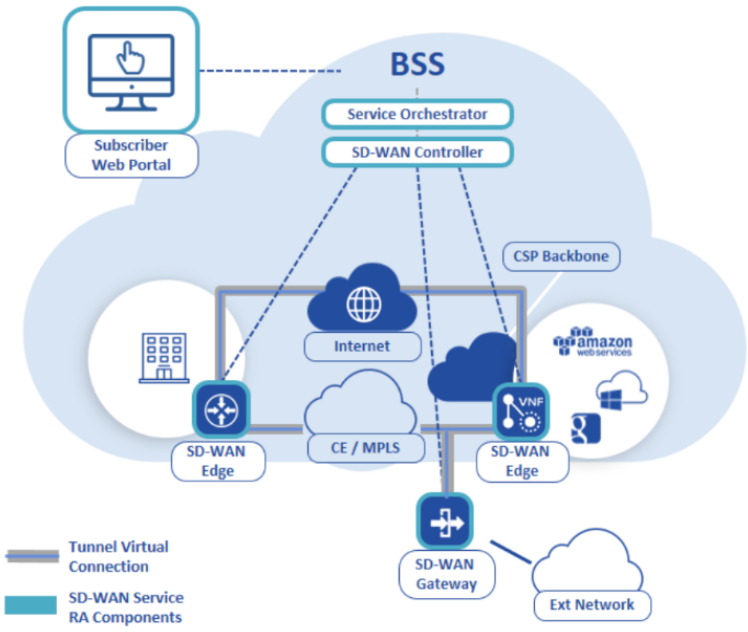
- Muracciole Vazquez, Andrés Jorge
- Bascopé Aparicio, Luis Leonardo

Tutor:

- Lentisco Sánchez, Carlos M.

1. Introducción:

Hoy en día la tecnología MPLS sobre redes WAN tradicionales ha ganado gran popularidad, sin embargo, estas suelen ser muy costosas para las empresas y su administración muy complicada y poco flexible. A raíz de esto, las redes SD-WAN (Software Defined-Wide Area Network) permiten reducir costes, no solamente de implementación y despliegue, sino también de administración y gestión. Los sistemas SD-WAN proporcionan una gran flexibilidad y reduce significativamente la complejidad de la red. SD-WAN permite al operador agregar, bajo demanda, tantas conexiones (públicas o privadas) como fuese necesario, para establecer la comunicación entre dos redes corporativas remotas, estableciendo criterios, reglas y políticas para dirigir los flujos de tráfico en base a criterios como el tipo de aplicación, o direcciones de origen y destino. SD-WAN se caracteriza por permitir el balanceo de tráfico entre diferentes tipos de conexiones (MPLS, Internet, etc.), seleccionar el camino de los datos dinámicamente por medio de políticas, soportar redes privadas virtuales (VPN) y proporcionar una interfaz sencilla para gestionar la WAN de forma global.



SD-WAN se basa en los conceptos y principios de SDN, los cuales incluyen el desacoplamiento del plano de control del plano de datos a la WAN. Esto significa la centralización del controlador de la totalidad de una red WAN corporativa, y administrarla y gestionarla por medio de una interfaz única. Administrar una WAN a través del software proporciona

beneficios útiles. En el pasado, realizar cambios en las configuraciones de red en las sucursales habría requerido la creación e instalación de configuraciones manuales y probablemente un técnico en el sitio para hacerlo; con SD WAN, por otro lado, la configuración en cada borde de la red se lo realiza de manera remota y todos equipos a la vez. Esto permite no solamente definir los parámetros de la red según la demanda sino, a partir de estas, brindar calidad de servicio y una mejor experiencia al usuario. Esta facilidad de despliegue, administración central y costes reducidos hacen de SD-WAN una opción atractiva para muchas empresas.

2. Objetivos

- Desarrollar un escenario de red con la herramienta de virtualización de redes Virtual Network over LinuX (VNX) que simule un entorno de red de área extensa "WAN" definida por software (SD-WAN) gestionada por el controlador RYU.
- Monitorizar el tráfico de la red SD WAN simulada en VNX, por medio del controlador RYU, utilizando "Traffic Monitor" y, a través de este, conseguir estadísticas que describan el comportamiento de la red.
- Estudiar las distintitas aplicaciones de RYU existentes que permiten extraer de la red SDN métricas de calidad de servicio.
- Analizar lo parámetros de calidad de servicio de Latencia, Ancho de Banda y perdida de paquetes, en el tráfico dl escenario
- Inspeccionar al tráfico con Wireshark.

3. Visión general del proyecto

El presente proyecto consiste en la elaboración de un escenario virtual, por medio de la herramienta de virtualización de redes "VNX". El escenario está compuesto por dos segmentos de red o "sucursales" las cuales están separadas por medio de una red WAN. Cada extremo de la red cuenta con dos enlaces redundantes al otro extremo de la misma. Los equipos involucrados son los siguientes:

- Dos ordenadores o "hosts", uno en cada extremo de la red. Estos son representados como contenedores Linux o "LXC", los cuales deben tener conectividad total entre ellos. Los hosts tienen acceso a la red por medio de conmutadores (uno en cada sucursal), las cuales permiten el flujo de tráfico y el intercambio de paquetes

- Un contenedor Linux “LXC” que haga el papel de una red MPLS dentro de la WAN.
- Un contenedor Linux “LXC” que haga el papel de una red Internet dentro de la WAN.
- Dos conmutadores que cumplan la función de extremos SD-WAN o “SD-WAN Edges”, cada uno a un extremo de la red WAN. Estos serán simulados como conmutadores OpenvSwtich. El plano de control en los conmutadores será desacoplado para ser gestionado por el controlador RYU.
- Un controlador RYU, el cual será ejecutado en el localhost y estará conectado a ambos conmutadores.

A continuación, se muestra el diagrama del escenario descrito:

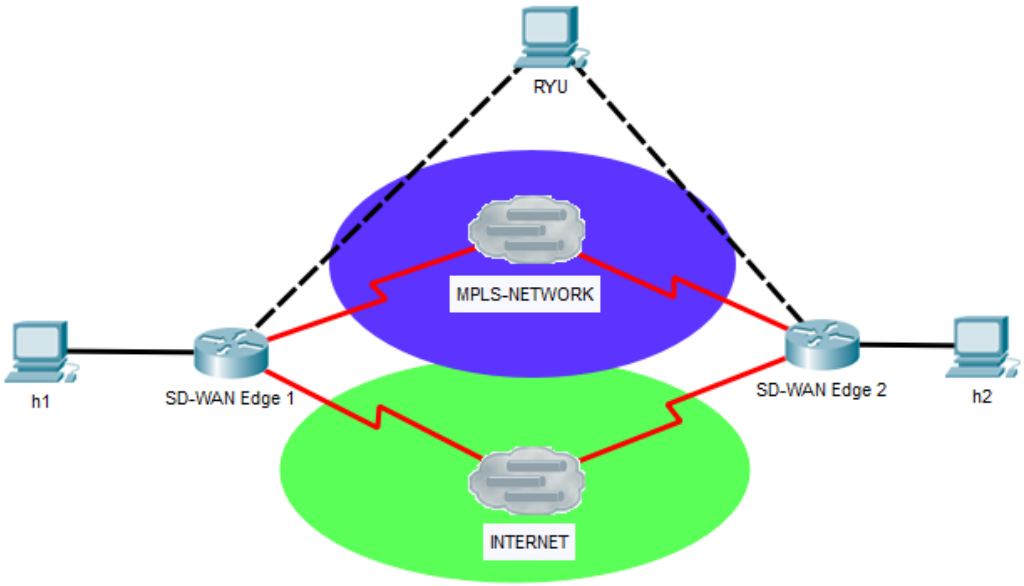


Figura 1: Topología de red deseada del Escenario.

A continuación, se explica en detalle cada una de las tecnologías utilizadas para el despliegue del escenario SD-WAN Virtualizado y la funcionalidad detallada de cada nodo y servicio clave para la implementación. Posteriormente, se detalla cómo realizar paso a paso la configuración y el despliegue del escenario con el controlador Ryu.

3.1.Tecnologías Implicadas:

3.1.1. VNX

Según la descripción en la wiki del departamento de ingeniería telemática de la Universidad Politécnica de Madrid, “*VNX es una herramienta de virtualización de código abierto de uso*

general diseñada para ayudar a construir modelos de pruebas de redes virtuales automáticamente". VNX es una herramienta útil para probar aplicaciones y servicios de red sobre escenarios virtuales complejos. Esta herramienta está compuesta de dos partes esenciales:

- Un fichero XML en el cual se describe el escenario de red virtual (lenguaje de especificación VNX).
- el programa VNX, que analiza la descripción del escenario en el fichero XML y construye y administra el escenario virtual en una máquina Linux.

Por medio de esta herramienta se simularán tanto los Hosts en ambas, y las redes WAN MPLS e Internet. Estas serán configuradas como contenedores Linux (LXC), cada una con un sistema de ficheros propio, el cual permitirá la gestión individual de los mismos.

3.1.2. OpenvSwitch:

OpenvSwitch es un software de código abierto, diseñado para ser utilizado como un switch virtual en entornos de servidores virtualizados. Es el encargado de reenviar el tráfico entre diferentes máquinas virtuales (VMs) en el mismo host físico y también reenviar el tráfico entre las máquinas virtuales y la red física. Este permite simular funcionalidades de un conmutador físico en un entorno virtual como el que se describe en el escenario del presente proyecto. Los SD-WAN Edges o "Extremos de la red" serán configurados por medio de OpenvSwitch.

3.1.3. RYU:

Ryu es un controlador de red de distribución abierta, para redes definidas por Software (SDN) diseñado para aumentar la agilidad de la red al facilitar la administración y la adaptación de cómo se maneja el tráfico. En general, el controlador SDN es el cerebro del entorno SDN, comunicando información a los conmutadores y enrutadores y de regreso al controlador por medio de OpenFlow.

Este es un framework de SDN escrito en el lenguaje Python que proporciona una API que facilita la creación de aplicaciones de administración y control de la red SD-WAN. El controlador que gestione el plano de control en los conmutadores será RYU, dado que hace posible la monitorización del tráfico en el presente proyecto.

A continuación, se muestra los logos de los tres elementos descritos.

Figura 2: Tecnologías implicadas en el proyecto.

4. Organización del proyecto

4.1.Periodos de trabajo

El presente proyecto fue realizado en tres periodos distintos, al final de los cuales se debía hacer una exposición del avance del proyecto. Dentro de estos periodos de trabajo se plantearon objetivos específicos. Estos tres periodos son los siguientes:

Periodo I:

Fechas: 6 de febrero – 14 de febrero:

Objetivos:

1. Planteamiento del problema
2. Diseño del Escenario

Periodo II:

Fechas: 17 de febrero – 6 de marzo:

Objetivos:

1. Estudio del controlador Ryu y sus características.

Periodo III:

Fechas: 7 de marzo – 20 de abril

Objetivos:

1. Estudio de los parámetros de calidad de servicio.
2. Presentación final del proyecto

4.2. Calendario de Trabajo

4.2.1. Febrero

FEBRERO														
LU	TAREA	HRS	MA	TAREA	HRS	MIE	TAREA	HRS	JU	TAREA	HRS	VIE	TAREA	HRS
						5	Elección del Proyecto	2	6	Planteamiento del problema y definición de Objetivos	3	7		
10			11			12	Estudio y prueba de escenarios en un entorno real de enrutamiento, utilizando OVS como Router	4	13	Reunión de trabajo y preparación de presentación	2	14	PRIMERA PRESENTACION DEL PROYECTO	2
17	DISEÑO DEL ESCENARIO INICIAL	4	18	DISEÑO DEL ESCENARIO DEFINITIVO	4	19	Definición de los parámetros QoS	3	20	**Reunión con Tutor**	0.5	21	Estudio y pruebas del controlador Ryu	2.5
24	Modificación de script Traffic Monitor para obtención de parámetros de QoS	4	25	Modificación de script Traffic Monitor para obtención de parámetros de QoS	4	26	Reunión de Trabajo	1	27	Modificación de script Traffic Monitor para obtención de parámetros de QoS	4	28		

El mes de febrero se caracterizó sobre todo por la organización del proyecto, planeación del cronograma de trabajo e investigación. El mismo está compuesto por el primer periodo, y la primera mitad del segundo. En el mismo se realizaron las siguientes tareas:

- Reuniones de trabajo: A lo largo del proyecto se hicieron reuniones periódicas entre los integrantes del grupo donde se definían objetivos a corto plazo, división de tareas y consultas. Por otra parte, se hacían reuniones breves con el tutor donde se evaluaban dudas y se definían objetivos a seguir.
- Diseño del escenario: Diseño del escenario en VNX. En una primera instancia se pensó en modelar las redes de MPLS, pero luego vimos que lo importante en el trabajo era medir parámetros de enlace y no tanto el tipo de red por los que viajan. Se fueron haciendo adaptaciones a lo largo del proyecto para llegar al objetivo final.
- Definición de parámetros para definición de QoS: Nos reunimos en primera instancia con el tutor y se definieron los parámetros ideales a monitorear. Estos fueron: Latencia, Ancho de banda y Perdida de Paquetes.
- Estudio de Ryu y sus herramientas: Estudio de diversas herramientas que proporciona Ryu por defecto, así como también aplicaciones de terceros publicadas en foros y

GitHub. También se estudiaron aplicaciones similares a la que nosotros precisábamos, de forma de encontrar ideas o funciones que sirvieran para poder lograr nuestros objetivos. Luego de encontrar algunas, se debatieron para determinar el alcance de cada una y poder decidir cuál cumplía con algunos de los requisitos que buscábamos. Se estudió el funcionamiento de aplicaciones en simultáneo (controlador, aplicación de QoS y aplicación de delay)

4.2.2. Marzo

MARZO														
LU	TAREA	Hrs	MAR	TAREA	Hrs	MIE	TAREA	Hrs	JU	TAREA	Hrs	VIE	TAREA	Hrs
2	Modificación de script Traffic Monitor para obtención de parámetros de QoS	4	3	Definición de Traffic Monitor como Controlador	3.5	4	Reunión de trabajo y preparación de presentación	3	5	Modificación de script Traffic Monitor para obtención de parámetros de QoS	4	6	SEGUNDA PRESENTACION DEL PROYECTO	2
9	Pruebas de Traffic Monitor	3	10	Pruebas de Traffic Monitor	1.5	11	Modificación de script Traffic Monitor para obtención de parámetros de QoS	4	12			13		
16	INICIO DEL PERIODO DE CAURENTENA: DEFINICION DE UN NUEVO PLAN DE TRABAJO POR MEDIO DE LA HERRAMIENTA TEAMS		17	Definición del Cálculo y modelado del BW	4	18	REUNION DE ASIGNACION DE PARAMETROS Y DELEGACION DE TAREAS	3	19			20	Definición del Calculo y modelado del BW	6
23			24			25			26	REUNION DE TRABAJO VIA TELEMATICA	3	27		
30	ESTUDIO DE LA DETERMINACION DE LA LATENCIA	6	31	ESTUDIO DE LA DETERMINACION DE LA LATENCIA	6									

El mes de marzo fue quizás el más complicado. Dada la situación de la cuarentena, en este se tuvo que replantear la organización del trabajo. En este comienza el tercer y último periodo de trabajo. El enfoque en este periodo fueron las siguientes tareas:

- Conexión Ryu con Traffic Control: Una vez elegido la aplicación Traffic Monitor fue necesario estudiar cómo conectarlo con nuestro escenario y el controlador. Para esto fue necesario estudiar el foro y manual e Ryu.
- Modificación de script Traffic Monitor para obtención de parámetros de QoS: Se hicieron adaptaciones de la aplicación Traffic Monitor para obtener parámetros relevantes como el ancho de banda. Este se configuró de forma que se imprima periódicamente en pantalla la información deseada. Este trabajo fue de los más tediosos ya que implicaba programar sobre un código ya implementado. A medida que se iban agregando funciones al código, este se iba probando y validando. Por otra parte, se iban guardando las diferentes versiones del mismo de forma de poder hacer un rollback en caso de ser necesario. El trabajo consistió en modificar funciones en lenguaje Python

4.2.3. Abril

ABRIL														
LU	TAREA	Hrs	MA	TAREA	Hrs	MIE	TAREA	Hrs	JU	TAREA	Hrs	VIE	TAREA	Hrs
						1			2	ESTUDIO DE LA DETERMINACION DE LA PERDIDA DE PAQUETES	6	3	ESTUDIO DE LA DETERMINACION DE LA PERDIDA DE PAQUETES	6
6			7			8			9			10	REUNION DE PREPARACION PARA LA PRESENTACION FINAL	2
13	REUNION DE PREPARACION PARA LA PRESENTACION FINAL	2	14	REDACCION DE LA MEMORIA	8	15	REDACCION FINAL DE LA MEMORIA	6	16	**REUNION FINAL CON EL TUTOR**	2	17	ELABORACION DE LA PRESENTACION FINAL	3
20	**PRESENTACION FINAL DEL PROYECTO**													

El mes de abril estuvo más enfocado al estudio de los parámetros de QoS:

- Modelado y cálculo del ancho de banda: El modelado del ancho de banda se lo realiza por medio una aplicación ya existente llamada “rest_qos.py”. La descripción técnica del trabajo ejecutado sobre la misma se describe en la siguiente sección.
- Cálculo del Delay: Se buscó en primera instancia una aplicación que realice este cálculo, pero no se obtuvo ninguna y que la implementación de una desde cero era muy

compleja. Se estudió teóricamente como debería ser la solución comparándola con otras adaptadas para otros tipos de controladores.

- Cálculo de la perdida de paquetes: Así como en el caso anterior, la determinación de la cantidad de paquetes perdidos no pudo realizarse con éxito, por los mismos motivos. Sin embargo, en la siguiente sección de describe de forma teórica cual debería ser el procedimiento de dicho cálculo.

El cronograma de trabajo, y las horas empleadas en la realización del proyecto fueron documentadas utilizando la herramienta “Trello”.

5. Implementación y despliegue del escenario

En el presente apartado se describe paso a paso la configuración y despliegue del escenario, y del controlador Ryu. Nótese que el presente reporte excluye el proceso de instalación de la herramienta VNX y del controlador Ryu, dado que los mismos ya vienen instalados en la OVA proporcionada por el docente de la asignatura SDNV. Sin embargo, dejamos la constancia de que, para iniciar la configuración y despliegue de este escenario, deben instalarse estas herramientas, siguiendo los pasos descritos en los enlaces siguientes:

VNX:

http://web.dit.upm.es/vnxwiki/index.php/Vnx-install

RYU:

https://osrg.github.io/ryu-book/en/html/installation_guide.html

5.1. Configuración del escenario

Como se mencionó anteriormente, el escenario será configurado por medio de la herramienta VNX. Para esto, se debe crear un fichero XML y en el mismo configurar cada uno de los elementos que describan el escenario deseado. Una lista de escenarios de referencia es encontrada en el directorio "examples" creada durante la instalación, y por un tema de orden, se recomienda crear el escenario del proyecto en la misma. La ruta de la misma es la siguiente:

```
$ cd /usr/share/vnx/examples
```

Se utilizó uno de los ficheros proporcionados durante la asignatura SDNV como referencia inicial en el diseño del escenario. Este está compuesto de dos partes esenciales:

- Contenedores Linux (LXC): estas no son máquinas virtuales como tal, sino más bien son entornos virtuales que ejecutan, cada uno, su propia instancia de un sistema operativo aislado. En el presente proyecto los contenedores Linux serán los cuatro hosts, así como las representaciones de las redes WAN: MPLS e Internet. Estos se configuran en el fichero XML por medio del denominativo "*vm*", especificando el tipo "*lxc*" y ejecutan dentro de sí una instancia de algún sistema operativo por medio de un "*root file system*". La configuración dentro de cada una puede variar de acuerdo a su función dentro de la red, por lo tanto, se describirá de manera específica cada elemento individual más adelante.

- OpenvSwitches: como se mencionó anteriormente estos cumplen el papel de switch virtual, y son los encargados de reenviar el tráfico entre diferentes máquinas virtuales (VMs) dentro del escenario. Los SD-WAN Edges o "Extremos de la red" serán configurados en el fichero XML por medio del denominativo "*net*" y especificando el modo "*openvswitch*". Los mismos serán controlados de forma remota, por medio de un controlador Ryu ubicado en el localhost de la máquina virtual.

A continuación, se procede a describir los elementos presentes en el escenario:

5.1.1. Hosts:

Los hosts son configurados con la letra "h" como nombre y un número secuencial el cual facilite la diferenciación entre ellos. Estos están definidos como contenedores Linux o "LXC". De acuerdo a los objetivos del escenario, el host "h1" fue configurado con una interfaz hacia el conmutador "SD-WAN_EDGE1"; y el host "h2" con una interfaz hacia el conmutador "SD-WAN_EDGE2".
Ambos hosts cuentan con direcciones MAC e IP diferentes. Estas están configuradas de acuerdo a la tabla de direccionamiento que se encuentra líneas más abajo.

Un ejemplo de la configuración de los hosts se muestra en la figura siguiente.

```
<vm name="h1" type="lxc" arch="x86_64">
  <filesystem type="cow">/usr/share/vnx/filesystems/rootfs_lxc_ubuntu64
  </filesystem>
  <mem>512M</mem>
  <if id="1" net="SD-WAN_EDGE1">
    <mac>00:00:00:00:01:01</mac>
    <ipv4>10.1.1.1/24</ipv4>
  </if>
  <route type="ipv4" gw="10.1.1.10">default</route>
</vm>
```

Figura 3: *Configuración de hosts*

Device	Interface	IP	SNM	Gateway	MAC	Network
h1	eth1	10.1.1.1	255.255.255.0	10.1.1.10	00:00:00:00:01:01	SD-WAN_EDGE1
h2	eth1	10.1.2.3	255.255.255.0	10.1.2.20	00:00:00:00:02:01	SD-WAN_EDGE1
MPLS	eth1	10.1.1.10	255.255.255.252		00:00:00:00:05:01	SD-WAN_EDGE1
MPLS	eth2	10.1.2.20	255.255.255.252		00:00:00:00:05:02	SD-WAN_EDGE2
INTERNET	eth1	10.1.1.30	255.255.255.252		00:00:00:00:06:01	SD-WAN_EDGE1
INTERNET	eth2	10.1.2.40	255.255.255.252		00:00:00:00:06:02	SD-WAN_EDGE2

Tabla 1: *Tabla de Direccionamiento*

5.1.2. Conmutadores OpenvSwitch – SD-WAN Edges:

Los extremos de la red o "SD WAN Edges" son configurados como switches virtuales, por medio de OpenvSwitch. A cada una se les asigna una dirección MAC diferente. En cada una se referencia el Socket del controlador (127.0.0.1:6633), el cual es el localhost. Así mismo se configura la versión de OpenFlow.
Un ejemplo de la configuración de los mismos se muestra a continuación:

```
<net name="SD-WAN_EDGE1" mode="openvswitch" hwaddr="00:00:00:00:07:00"
 controller="tcp:127.0.0.1:6633" of_version="OpenFlow13"
  fail_mode="secure" />
```

Figura 4: *Configuración de los OVS*

5.1.3. Redes WAN

Así como los hosts, las redes WAN (MPLS e Internet) son configuradas como contenedores Linux (LXC). Cada uno de ellos cuenta con una interfaz a cada uno de los conmutadores SD-WAN Edge; cada una con su respectiva dirección IP y MAC. Nótese que la dirección IP de estas interfaces, corresponde a las puertas de enlace por defecto o "Default Gateways" en los hosts. Así mismo, al arrancar inicialmente estos contenedores, se ejecuta independientemente en cada uno de ellos una línea de código la cual permite el enrutamiento IP o "IP Forwarding". Este código es el siguiente:

```
<exec seq="on_boot" type="verbatim">
    sudo sysctl net.ipv4.ip_forward=1
</exec>
```

Figura 5: *Habilitar el enrutamiento IP*

El IP Forwarding es un proceso utilizado para determinar qué ruta se puede enviar un paquete. Un ejemplo de la configuración de estos nodos se muestra a continuación:

```
<vm name="MPLS_NETWORK" type="lxc" arch="x86_64">
    <filesystem type="cow">/usr/share/vnx/filesystems/rootfs_lxc_vyos64
    </filesystem>
    <if id="1" net="SD-WAN_EDGE1" >
      <mac>00:00:00:00:05:01</mac>
      <ipv4>10.1.1.10/24</ipv4>
    </if>
    <if id="2" net="SD-WAN_EDGE2" >
      <mac>00:00:00:00:05:02</mac>
      <ipv4>10.1.2.20/24</ipv4>
    </if>
    <exec seq="on_boot" type="verbatim">
        sudo sysctl net.ipv4.ip_forward=1
    </exec>
  </vm>
```

Figura 6: *Configuración de nodos WAN*

La herramienta VNX tiene muchas funciones que la hacen cómoda de usar para el usuario. Una de las opciones más útiles es la capacidad de mostrar un mapa del escenario configurado en un fichero XML específico. Esto se realiza por medio del siguiente comando:

```
$ sudo vnx -f Lab_MinProy.xml -v --show-map
```

El resultado de esta sentencia, es la imagen que vemos líneas más abajo. Nótese que la misma representa a cabalidad la topología deseada, descrita en la Figura 1: “Topología de red deseada del escenario”.

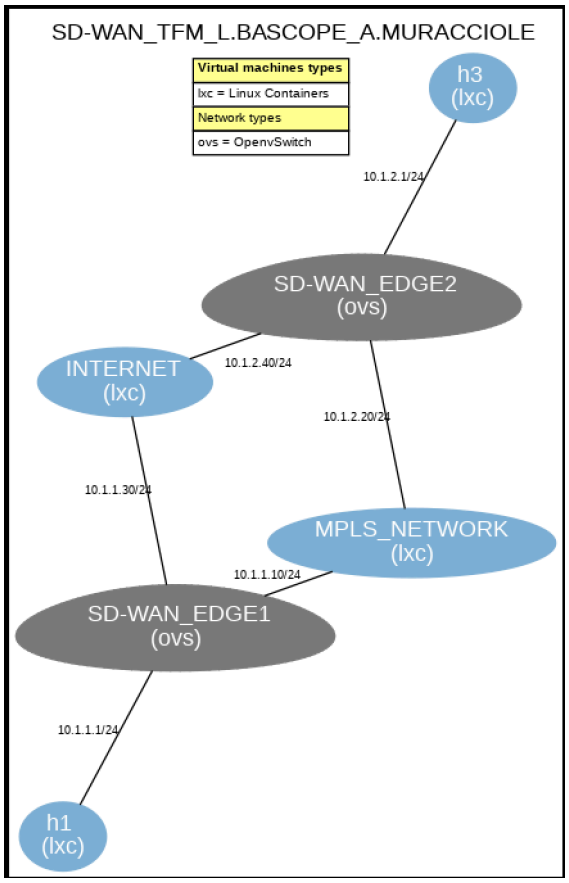


Figura 7: *Topología de red configurada del escenario*

Finalmente, para desplegar el entorno configurado en el fichero XML, es necesario ejecutar la siguiente sentencia:

```
$ sudo vnx -f Lab_MinProy.xml -v --create
```

Inmediatamente, si no existen problemas en la configuración, el escenario será creado y estará operacional en cuestión de segundos.

6. El controlador Ryu

Una vez diseñado e implementado el escenario de red, se prosiguió con la configuración del controlador Ryu. Investigando acerca de las aplicaciones del mismo, se llegó a la conclusión de que lo que más se adaptaba al objetivo del presente proyecto sería la utilización de la aplicación "Traffic Monitor", la cual viene embebida dentro del archivo "simple_monitor_13.py" originario del controlador.

De acuerdo a los objetivos planteados, el controlador debe ser capaz de monitorizar la red en tiempo real, y mostrar en pantalla estadísticas de tráfico específicas. Los parámetros deseados que se desean calcular son:

- Ancho de Banda (BW)
- Perdida de Paquetes (Packet Loss)
- Latencia (Delay)

Habiendo definido estos parámetros, es que pasamos a analizar uno a uno los mismos.

6.1. Perdida de paquetes (Packet Loss)

En toda red de comunicaciones existe la posibilidad de que parte de los paquetes que circulan a través de esta se pierdan por distintos motivos como ser: hardware dañado, cuellos de botella, descarte de paquete en colas de congestión en los conmutadores, etc. La pérdida de paquetes puede o no ser un problema, dependiendo del tipo de paquete que se pierda y la cantidad de paquetes perdidos. Sea como fuere, esta afecta al rendimiento de la red por lo que conocer este dato es fundamental para determinar el grado de incidencia que tiene la misma sobre la red, y minimizarlo.

Como se mencionó anteriormente, la aplicación “Traffic Monitor” es una aplicación propia de Ryu la cual, por medio de peticiones a la API, es capaz monitorizar parámetros de la red en tiempo real y procesarlos de forma en que sean fáciles de interpretar para el usuario. Su principal funcionalidad es imprimir en pantalla de forma periódica dos tablas:

- Tablas de flujo o “flow table” y
- Tabla de puertos o “port table”

Por medio de estas dos tablas, se procederá a calcular la cantidad de paquetes perdidos.

La tabla de flujo es aquella que va registrando la cantidad de paquetes que van cumpliendo con determinadas reglas. En este caso fue necesario adaptarla para que devuelva la cantidad de paquetes con un determinada MAC de origen y MAC de destino como se puede ver en la siguiente imagen:

datapath	in-port	eth-src	eth-dst	packets
0000000000000700	1	00:00:00:00:01:01	00:00:00:00:05:01	77
0000000000000700	2	00:00:00:00:05:01	00:00:00:00:01:01	78
0000000000000800	1	00:00:00:00:03:01	00:00:00:00:05:02	78
0000000000000800	2	00:00:00:00:05:02	00:00:00:00:03:01	77

Figura 8: *Tabla de Flujo*

A medida que los paquetes entrantes o salientes coincidan con alguna regla de dicha tabla, el valor del contador (packet) se va incrementando.
De una forma similar sucede en la Tabla de Puertos. Esta muestra los estados de los puertos de cada conmutador, entre los que destacan los campos "rx-pkts" y "tx-pkts". Estos son contadores que indican la cantidad de paquetes recibidos y transmitidos respectivamente por cada puerto de cada conmutador.

datapath	port	rx-pkts	tx-pkts	port-bw(Kbps)	port-speed(Kbps)	port-freebw(Kbps)	port-state	link-state
1792	1	105	105	100000	0.0	100000.0	up	Live
1792	2	104	105	100000	0.0	100000.0	up	Live
1792	3	29	27	100000	0.0	100000.0	up	Live
2048	1	103	105	100000	0.0	100000.0	up	Live
2048	2	104	105	100000	0.0	100000.0	up	Live
2048	3	28	27	100000	0.0	100000.0	up	Live

Figura 9: *Tabla de puertos*

Una vez entendido el funcionamiento e importancia de cada una de ellas, es que se puede determinar la cantidad de paquetes perdidos.
Para ello es necesario visualizar ambas tablas en un momento cualquiera. A modo de ejemplo se toman las tablas según se indica en la figura 10, a continuación:

datapath	in-port	eth-src	eth-dst	packets
0000000000000700	1	00:00:00:00:01:01	00:00:00:00:05:01	77
0000000000000700	2	00:00:00:00:05:01	00:00:00:00:01:01	78
0000000000000800	1	00:00:00:00:03:01	00:00:00:00:05:02	78
0000000000000800	2	00:00:00:00:05:02	00:00:00:00:03:01	77

datapath	port	rx-pkts	tx-pkts	port-bw(Kbps)	port-speed(Kbps)	port-freebw(Kbps)	port-state	link-state
1792	1	105	105	100000	0.0	100000.0	up	Live
1792	2	104	105	100000	0.0	100000.0	up	Live
1792	3	29	27	100000	0.0	100000.0	up	Live
2048	1	103	105	100000	0.0	100000.0	up	Live
2048	2	104	105	100000	0.0	100000.0	up	Live
2048	3	28	27	100000	0.0	100000.0	up	Live

Figura 10: *Flow Table y Port Table en un momento dado*

En este momento es que se hace un ping entre host “h1”, cuya MAC es “00:00:00:00:01:01” y el host “h2” cuya MAC es “00:00:00:00:03:01”. De esta forma es que, en la próxima actualización de tablas, estas se ven de la siguiente forma:

```
   datapath         in-port      eth-src             eth-dst         packets
----------------   -------- -----------------   -----------------  ---------
0000000000000700         1   00:00:00:00:01:01   00:00:00:00:05:01       78
0000000000000700         2   00:00:00:00:05:01   00:00:00:00:01:01       79
0000000000000800         1   00:00:00:00:03:01   00:00:00:00:05:02       79
0000000000000800         2   00:00:00:00:05:02   00:00:00:00:03:01       78


datapath  port   rx-pkts    tx-pkts  port-bw(Kbps)  port-speed(Kbps) port-freebw(Kbps)  port-state  link-state
--------  ----  ---------  --------- -------------  ---------------  -----------------  ----------  ----------
    1792     1        106        106     100000              0.1           99999.9          up        Live
    1792     2        105        106     100000              0.1           99999.9          up        Live
    1792     3         29         27     100000              0.0          100000.0          up        Live
    2048     1        104        106     100000              0.1           99999.9          up        Live
    2048     2        105        106     100000              0.1           99999.9          up        Live
    2048     3         28         27     100000              0.0          100000.0          up        Live
```

Figura 11: *Flow Table y Port Table luego de realizar un ping entre h1 y h2*

De las dos figuras anteriores se llega a las siguientes conclusiones:

- El ICMP request entra por la interfaz 1 del conmutador 700 (o 1792) y sale por la interfaz 2 del conmutador 700 (o 1792). Esto corresponde a la entrada 1 de la primera tabla (flow table).
- El ICMP request entra por la interfaz 2 del conmutador 800 (o 2048) y sale por la interfaz 1 del conmutador 800 (o 2048) llegando hasta el host 2. Esto corresponde a la entrada 4 de la primera tabla (flow table).
- El ICMP reply entra por la interfaz 1 del conmutador 800 (o 2048) y sale por la interfaz 2 del conmutador 800 (o 2048). Esto corresponde a la entrada 3 de la primera tabla (flow table).
- El ICMP reply entra por la interfaz 2 del conmutador 700 (o 1792) y sale por la interfaz 1 del conmutador 700 (o 1792) llegando hasta el host 1. Esto corresponde a la entrada 2 de la primera tabla (flow table).

Por otra parte, de la tabla de puertos o “port table” (la de abajo), se aprecia que:

- La cantidad de paquetes enviados y recibidos por la interfaz 1 del conmutador 700 (o 1792) era de 105 y 105 respectivamente y que luego del ping pasó a 106 y 106.
- La cantidad de paquetes enviados y recibidos por la interfaz 2 del conmutador 700 (o 1792) era de 104 y 105 respectivamente y que luego del ping pasó a 105 y 106.

- La cantidad de paquetes enviados y recibidos por la interfaz 1 del conmutador 800 (o 2048) era de 103 y 105 respectivamente y que luego del ping pasó a 104 y 106.
- La cantidad de paquetes enviados y recibidos por la interfaz 2 del conmutador 800 (o 2048) era de 104 y 105 respectivamente y que luego del ping pasó a 105 y 106.

Por lo tanto, ante un único ping, todos los contadores por donde pasó este paquete se incrementaron en uno, lo cual es coherente y para este caso en particular se dice que la pérdida es de 0%.

Conociendo la cantidad de paquetes salientes por un conmutador, las direcciones de origen y destino y la cantidad de paquetes recibidos en el otro conmutador es que se calcula la pérdida de paquetes (Packet Loss). En caso de existir diferencias entre ambas tablas, se puede saber, no solo que hubo pérdida de paquetes, sino también la cantidad de paquetes que se perdieron. El valor del Packet Loss se puede definir por medio de la siguiente ecuación:

$$Packet\ Loss = \frac{\#tx\_pkts - \#rx\_pkts}{\#tx\_pkts} \times 100$$

Figura 12: *Ecuación para determinar el Packet Loss*

6.2. Ancho de Banda (BW)

El cálculo del ancho de banda es un poco más complejo que el anterior ya que este no se consigue de forma explícita mediante la aplicación de "Traffic Monitor" que viene por defecto; en este caso es necesario modificarla. Afortunadamente Ryu es un controlador de distribución abierta y existe una gran comunidad de usuarios que ponen sus aportes a disposición del público. Gracias a esto no fue necesario crear un código nuevo de aplicación, sino que se adaptó el código de la aplicación con otra ya existente, que si contaba con el cálculo necesario para implementarlo a la "Port Table" mencionada anteriormente.

Lo primero fue definir dos nuevas comunas. La primera de estas mostraba el ancho de banda físico del canal y el tráfico cursado en tiempo real y la segunda, el ancho de banda disponible. Esta segunda surge de restar el consumo actual al valor del ancho de banda físico del mismo.

Para demostrar que este se comporta correctamente, se procedió a definir, por medio de políticas de calidad de servicio, un ancho de banda de 100 Mbps en cada enlace. Para ello fue necesario utilizar otra aplicación propia de Ryu capaz de definir estos umbrales.

6.2.1. Modelado del ancho de banda por medio de políticas de calidad de servicio

La aplicación propia de Ryu utilizada para modelar el ancho de banda y controlar el flujo lleva como nombre "**rest_qos.py**". Por medio de esta que podemos moldear el ancho de banda por interfaz de cada switch virtual y limitar el ancho de banda al trafico puerto TCP/UDP, si fuese necesario. Para ejecutar exitosamente esta aplicación, se necesitan dos datos principales, los cuales son:

- el datapath ID (dpid) de cada OVS y
- el nombre de puerto (port_name) de cada interfaz.

6.2.1.1. Establecer la cola de los conmutadores OVS

El primer paso en el modelado del ancho de banda, es establecer la cola en cada uno de los conmutadores OVS. Será dentro de esta cola donde se definirán las reglas que modelen el flujo del tráfico.
Es necesario conocer los datapath ID de cada uno de los Switches OVS. Este valor es mostrado en pantalla durante el inicio del controlador, como se muestra a continuación:

```
[QoS][INFO] dpid=0000000000000700: Join qos switch.
[QoS][INFO] dpid=0000000000000800: Join qos switch.
```

Figura 13: *Datapath ID*

Con estos valores, configuramos la dirección de la base datos de cada OVS o "ovsdb_addr" para acceder a las mismas, con las siguientes peticiones CURL en el controlador:

```
curl -X PUT -d '"tcp:127.0.0.1:6632"'
http://localhost:8080/v1.0/conf/switches/0000000000000700/ovsdb_addr

curl -X PUT -d '"tcp:127.0.0.1:6632"'
http://localhost:8080/v1.0/conf/switches/0000000000000800/ovsdb_addr
```

Una vez definidas las direcciones de las bases de datos para cada OVS, se debe ejecutar la configuración de la cola en cada una de las interfaces de cada OVS. Por tal motivo es necesario conocer los nombres de estas interfaces. Este valor lo conseguimos con el siguiente comando:

```
$ sudo ovs-vsctl show
```

El resultado del mismo es el siguiente:

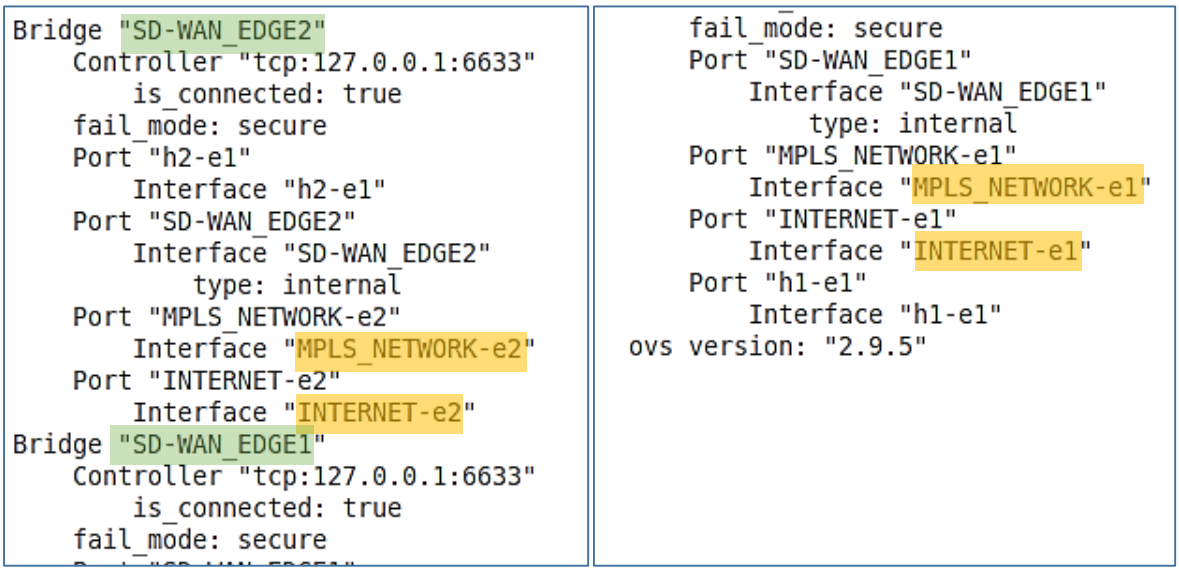


Figura 14: *Obtención de los nombres de los puertos*

Habiendo obtenido el nombre de los puertos, configuramos las colas para cada uno de los Switches OVS. Esto se realiza por medio de las peticiones CURL a las bases de datos de los dos OVS:

```
curl -X POST -d '{"port_name": "MPLS_NETWORK-e1", "type": "linux-htb", "queues": [{"max_rate": "100000000"}]}' http://localhost:8080/qos/queue/0000000000000700

curl -X POST -d '{"port_name": "INTERNET-e1", "type": "linux-htb", "queues": [{"max_rate": "100000000"}]}' http://localhost:8080/qos/queue/0000000000000700

curl -X POST -d '{"port_name": "MPLS_NETWORK-e2", "type": "linux-htb", "queues": [{"max_rate": "100000000"}]}' http://localhost:8080/qos/queue/0000000000000800

curl -X POST -d '{"port_name": "INTERNET-e2", "type": "linux-htb", "queues": [{"max_rate": "100000000"}]}' http://localhost:8080/qos/queue/0000000000000800
```

Como se puede apreciar, se crea una regla por cada una de las interfaces, en cada uno de los dos conmutadores; es decir se añaden cuatro reglas. Las mismas limitan el ancho de banda de 100Mbps a un valor máximo de 100Mbps, para cada interfaz.

6.2.2. Verificación de la configuración en los Switches OVS

Una vez habiendo modelado ancho de banda de cada interfaz a 100 Mbps, se procedió a la verificación de la configuración por medio de un "iperf". Iperf es una herramienta que permite crear flujos entre un origen y un destino, con el fin de probar el rendimiento de la red. El iperf se realizó entre los hosts "h1" y "h2". El flujo creado en la prueba fue de 50 Mbps (la mitad del máximo de la capacidad) para ver que efectivamente el ancho de banda disponible pasaba a ser la mitad.
Para realizar esta prueba, en el host "h2" se ejecutó el comando siguiente:

```
> iperf -s -i 1
```

Esto haría que el host "h2" se quede escuchando a la recepción de cualquier tráfico. Así mismo, se ejecutó el siguiente comando el host "h1":

```
> iperf -c 10.1.2.3 -i 1 -u -b 50000000
```

Al ejecutar este comando, se crea un flujo desde el host "h1" al dispositivo cuya dirección IP sea "10.1.2.3", es decir ""h2. Este flujo creado tiene un ancho de banda de 50Mbps, lo que debería ocupar la mitad de los recursos de la red.
El resultado de la prueba se puede apreciar a continuación:

```
   datapath        in-port       eth-src             eth-dst         packets
----------------   -------- -----------------   -----------------  ---------
0000000000000700         1   00:00:00:00:01:01   00:00:00:00:05:01   113807
0000000000000700         2   00:00:00:00:05:01   00:00:00:00:01:01       83
0000000000000800         1   00:00:00:00:03:01   00:00:00:00:05:02       84
0000000000000800         2   00:00:00:00:05:02   00:00:00:00:03:01   113808

datapath  port   rx-pkts    tx-pkts  port-bw(Kbps)  port-speed(Kbps) port-freebw(Kbps)  port-state  link-state
--------  ----  ---------  --------- -------------  ---------------  -----------------  ----------  ----------
    1792     1     114907        113     100000             0.0           100000.0          up        Live
    1792     2        111     114907     100000          51429.7            48570.3          up        Live
    1792     3         31         29     100000             0.0           100000.0          up        Live
    2048     1        112     114908     100000          51427.7            48572.3          up        Live
    2048     2     114907        113     100000             0.0           100000.0          up        Live
    2048     3         30         30     100000             0.0           100000.0          up        Live
```

Figura 13: *Flow Table y Port Table luego de realizar un iperf entre h1 y h2 UDP de 50Mbps*

Podemos ver en la columna "*port-bw(Kbps)*" de la Port Table, el ancho de banda del canal, y como este coincide con el ancho de banda configurado. Además, se aprecia en la columna "*port-freebw(Kbps)*", como el ancho de banda disponible en el canal disminuye a la mitad con cada flujo del iperf que pasa por él.

Haciendo una prueba de mayor estrés y haciendo un iperf con la opción *"-P 2"* se pueden simular dos flujos iperf UDP de 50 Mbps cada uno n simultáneo (utiliza dos puertos distintos). Haciendo esto se ve que el ancho de banda disponible baja a 0.

datapath	in-port	eth-src	eth-dst	packets
0000000000000700	1	00:00:00:00:01:01	00:00:00:00:05:01	278384
0000000000000700	2	00:00:00:00:05:01	00:00:00:00:01:01	91
0000000000000800	1	00:00:00:00:03:01	00:00:00:00:05:02	92
0000000000000800	2	00:00:00:00:05:02	00:00:00:00:03:01	278385

datapath	port	rx-pkts	tx-pkts	port-bw(Kbps)	port-speed(Kbps)	port-freebw(Kbps)	port-state	link-state
1792	1	280706	121	100000	0.0	100000.0	up	Live
1792	2	119	280706	100000	102847.3	0.0	up	Live
1792	3	32	30	100000	0.1	99999.9	up	Live
2048	1	120	280708	100000	102864.6	0.0	up	Live
2048	2	280706	123	100000	0.1	99999.9	up	Live
2048	3	31	31	100000	0.1	99999.9	up	Live

Figura 14: *Flow Table y Port Table luego de realizar dos iperf simultáneos entre h1 y h2 UDP de 50Mbps cada uno*

Lo mismo sucede si se hacen dos iperf de 40 Mbps cada uno. En ancho de banda disponible queda en 20 Mbps aproximadamente como se ve a continuación:

datapath	in-port	eth-src	eth-dst	packets
0000000000000700	1	00:00:00:00:01:01	00:00:00:00:05:01	710518
0000000000000700	2	00:00:00:00:05:01	00:00:00:00:01:01	109
0000000000000800	1	00:00:00:00:03:01	00:00:00:00:05:02	109
0000000000000800	2	00:00:00:00:05:02	00:00:00:00:03:01	710518

datapath	port	rx-pkts	tx-pkts	port-bw(Kbps)	port-speed(Kbps)	port-freebw(Kbps)	port-state	link-state
1792	1	711066	139	100000	0.0	100000.0	up	Live
1792	2	137	711066	100000	82291.5	17708.5	up	Live
1792	3	32	31	100000	0.0	100000.0	up	Live
2048	1	137	711068	100000	82291.6	17708.4	up	Live
2048	2	711065	140	100000	0.1	99999.9	up	Live
2048	3	31	31	100000	0.0	100000.0	up	Live

Figura 15: *Flow Table y Port Table luego de realizar dos iperf simultáneos entre h1 y h2 UDP de 40Mbps cada uno*

La inexactitud se debe a que el programa va haciendo un promedio del estado de los enlaces de los últimos 10 segundos. En caso de dejar durante varios segundos un iperf ejecutándose, se puede ver que se va acercando a este valor.

6.3. Latencia (Delay)

Se entiende por latencia, el tiempo que tarda en trasmitirse un paquete por la red desde un origen hasta su destino. El monitoreo de la latencia en la red consiste en crear un paquete el cual será usado como una sonda. Tras generar este paquete, el controlador, por medio de un mensaje "PacketOut", solicita al Switch OVS que transmita el mismo por medio de una de las interfaces conectadas a él que conduzcan hacia el destino del paquete. Finalmente, el switch OVS, ubicado al otro lado de la red recibe este paquete, y le envía un mensaje "PacketIn" al controlador para comunicar el estado del paquete. La siguiente figura Ilustra este proceso.

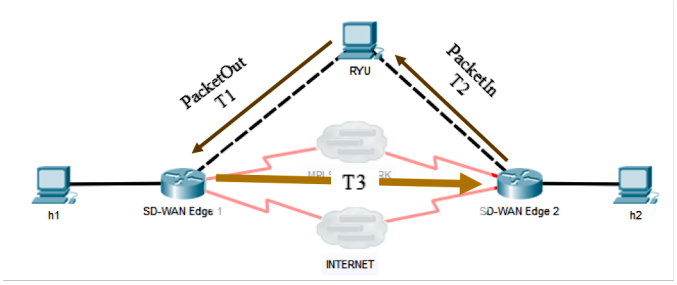


Figura 16: Proceso de medición de la latencia

El tiempo que nos interesa conocer, que representa el dato de la latencia en la red es el correspondiente a "T3" en la Figura 16. Este representa el tiempo que tarda un paquete en llega de un switch al otro.

Desde la generación de un paquete en el origen hasta la recepción del mensaje PacketIn en el controlador, el tiempo TOTAL es determinado de la siguiente manera:

$$Ttotal = T1 + T2 + T3$$

Donde:

- T1: Tiempo que demora el mensaje PacketOut desde el controlador hasta el Switch
- T2: Tiempo que demora el mensaje Packet desde Switch hasta el controlador

Para calcular los valores de T1 y T2, se debe tener en cuenta el RTT (Round Trip Time) de los mensajes "pregunta-respuesta" entre el switch y el controlador. Este tiempo se expresa de la siguiente manera:

$$T1 = 0.5 * (Tb - Ta)$$

En la ecuación anterior se observan dos nuevas Variables:

- ✓ Ta: instante en que se envía el paquete de prueba. En este caso es una solicitud OpenFlow de tipo "ports-stats-request".
- ✓ Tb: instante en que se recibe la respuesta al mensaje que es de tipo "port-stats-received".

Como consecuencia, el valor de la latencia que buscamos es:

$$T3 = Ttotal - T1 - T2$$

6.3.1. Implementación de latencia al controlador

Habiendo analizado el algoritmo, para que el valor de la latencia se vea contemplado en las estadísticas del controlador, es necesario crear una nueva columna en el controlador que almacene el valor de la latencia calculado y lo muestre por pantalla.

El cálculo de la latencia se la realiza en el controlador de la siguiente manera:

```
def _handle_portstats_received (event):

   global start_time, sent_time1, sent_time2, received_time1, received_time2, src_dpid, dst_dpid,OWD1,OWD2

   received_time = time.time() * 1000 - start_time

   #measure T1

   if event.connection.dpid == src_dpid:
     OWD1=0.5*(received_time - sent_time1)
     #print "OWD1: ", OWD1, "ms"
   #measure T2
   elif event.connection.dpid == dst_dpid:
     OWD2=0.5*(received_time - sent_time1)
     #print "OWD2: ", OWD2, "ms"
```

En el código de arriba, vemos como se calculan los valores de T1 y T2, por medio de los cuales se puede obtener eventualmente el valor de la latencia o T3:

```
def _handle_PacketIn (event):

  global start_time,OWD1,OWD2


  packet = event.parsed
  #print packet
  received_time = time.time() * 1000 - start_time

  if packet.type==0x5577 and event.connection.dpid==dst_dpid:
    c=packet.find('ethernet').payload
    d,=struct.unpack('!I', c)
    #obtain T3
    print "delay:", received_time - d - OWD1-OWD2, "ms"
    f2.write("Packet-In,%s,%s,%s\n"%(src_dpid,dst_dpid,(received_time - d - OWD1-OWD2)))
    f2.flush()
```

Hecho esto, el resultado en el controlador debería ser el siguiente:

```
delay: 16.1680908203 ms
delay: 16.5806884766 ms
delay: 16.4718017578 ms
delay: 15.6439208984 ms
delay: 16.4467773438 ms
```

Figura 17: *Resultado de la latencia*

7. Conclusiones y Recomendaciones

- ✓ El diseño original del escenario contaba con cuatro ordenadores: dos de cada extremo. Sin embargo, por temas de objetividad con respecto al flujo de datos, y la facilidad de interpretación de los resultados, se optó por tener solo uno en cada lado. Fuera de eso, el escenario final se mantuvo de acuerdo a lo planteado en los objetivos. El código del mismo se encuentra en el Anexo 1.
- ✓ En cuanto al ancho de banda, este fue posible calcularlo a partir de modificaciones realizadas en el Traffic Monitor de forma que se puede saber en tiempo real el ancho de banda utilizado y disponible en tiempo real. En este caso tuvimos el inconveniente que haciendo la consulta a la API, esta devolvía que los canales eran de 10 Gbps, cuando haciendo un iperf, este daba de más de 40 Gbps. Por lo tanto, la forma que tuvimos de limitar el canal era simulando uno de 100 Mbps (utilizando la aplicación "qos_rest.py") y fijando estáticamente a nivel de configuración que el canal era de 100 Mbps. Se recomienda a futuro poder encontrar la forma de poder determinar de forma más concreta el valor real del ancho de banda del canal para que el programa sea aún más escalable ya que en la vida real no todos los enlaces de una red son iguales (en este caso de 100 Mbps)
- ✓ La pérdida de paquetes (Packet Loss) se calculó siguiendo las recomendaciones de la guía misma de Ryu, en la sección el Traffic Monitor. Si bien la idea en un principio fue poder imprimir en pantalla el valor específico porcentual de la perdida de paquetes, se vio que eso era muy complejo ya que era necesario diseñar un bloque de código en nuestro programa que cruce la información de la Flow Table y la Port Table, y en función de esta realice la cuenta de paquetes perdidos y lo devuelva en pantalla. Esto vimos que era altamente costoso y lo comprobamos en foros donde hacían mención de lo mismo. Como recomendación para un trabajo a futuro podría ser justamente el trabajo de a partir de lo que se devuelve en ambas tablas, hacer la función que las vincule entre sí (por ejemplo, utilizando el dpid, dirección de origen y dirección de destino, y evalúe la cantidad de paquetes enviados y recibidos y realice el cálculo.
- ✓ En cuanto al cálculo de la latencia, el algoritmo para obtener el valor es relativamente sencillo, sin embargo, la dificultad radica en modificar el código del controlador para implementarlo. Se recomienda que en un trabajo a futuro este cálculo pueda ser implementado.

ANEXOS

1. Escenario: fichero XML

```
<?xml version="1.0" encoding="UTF-8"?>

<!--

~~~~~~~~~~~~~~~~~~~~~~

VNX ESCENARIO DE LA ASIGNATURA MPRO - PROYECTO:
MONITORIZACIÓN DE TRÁFICO EN REDES WAN
DEFINIDAS POR SOFTWARE (SD-WAN) MEDIANTE CONTROLADOR RYU

~~~~~~~~~~~~~~~~~~~~~~

Name:        SD-WAN_TFM

Description:
!!!!!!!!!!ESCENARIO DE MPRO: SD-WAN!!!!!!!!!!!!!!!

MASTER UIVERSITARIO EN INGENIERIA DE REDES Y SERVICIOS TELEMATICOS
Universidad Politecnica de Madrid
SPAIN

-->

<vnx xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="/usr/share/xml/vnx/vnx-2.00.xsd">
  <global>
    <version>2.0</version>
    <scenario_name>SD-WAN_TFM_L.BASCOPE_A.MURACCIOLE</scenario_name>
    <automac offset="5"/>
    <vm_mgmt type="none" />
    <vm_defaults>
        <console id="0" display="no"/>
        <console id="1" display="yes"/>
    </vm_defaults>
  </global>
  <net name="SD-WAN_EDGE1" mode="openvswitch" hwaddr="00:00:00:00:07:00" controller="tcp:127.0.0.1:6633"
    of_version="OpenFlow13" fail_mode="secure" />
  <net name="SD-WAN_EDGE2" mode="openvswitch" hwaddr="00:00:00:00:08:00" controller="tcp:127.0.0.1:6633"
    of_version="OpenFlow13" fail_mode="secure" />

  <!-- NODES -->
  <vm name="h1" type="lxc" arch="x86_64">
    <filesystem type="cow">/usr/share/vnx/filesystems/rootfs_lxc_ubuntu64</filesystem>
    <mem>512M</mem>
    <if id="1" net="SD-WAN_EDGE1">
      <mac>00:00:00:00:01:01</mac>
      <ipv4>10.1.1.1/24</ipv4>
    </if>
    <route type="ipv4" gw="10.1.1.10">default</route>
  </vm>
  <vm name="h2" type="lxc" arch="x86_64">
    <filesystem type="cow">/usr/share/vnx/filesystems/rootfs_lxc_ubuntu64</filesystem>
    <if id="1" net="SD-WAN_EDGE2" >
      <mac>00:00:00:00:03:01</mac>
      <ipv4>10.1.2.3/24</ipv4>
    </if>
    <route type="ipv4" gw="10.1.2.20">default</route>
  </vm>
```

```
  <vm name="MPLS_NETWORK" type="lxc" arch="x86_64">
    <filesystem type="cow">/usr/share/vnx/filesystems/rootfs_lxc_ubuntu64</filesystem>
    <if id="1" net="SD-WAN_EDGE1" >
      <mac>00:00:00:00:05:01</mac>
      <ipv4>10.1.1.10/24</ipv4>
    </if>
    <if id="2" net="SD-WAN_EDGE2" >
      <mac>00:00:00:00:05:02</mac>
      <ipv4>10.1.2.20/24</ipv4>
    </if>
    <exec seq="on_boot" type="verbatim">
        sudo sysctl net.ipv4.ip_forward=1
    </exec>
  </vm>
 <vm name="INTERNET" type="lxc" arch="x86_64">
    <filesystem type="cow">/usr/share/vnx/filesystems/rootfs_lxc_ubuntu64</filesystem>
    <if id="1" net="SD-WAN_EDGE1" >
      <mac>00:00:00:00:06:01</mac>
      <ipv4>10.1.1.30/24</ipv4>
    </if>
    <if id="2" net="SD-WAN_EDGE2" >
      <mac>00:00:00:00:06:02</mac>
      <ipv4>10.1.2.40/24</ipv4>
    </if>
    <exec seq="on_boot" type="verbatim">
        sudo sysctl net.ipv4.ip_forward=1
    </exec>
  </vm>
</vnx>
```