Pruebas de escala de VPNs capa 2 y 3 para la implementación legada basada en MPLS

Gustavo López
gustavo.lopez@fing.edu.uy

Facultad de Ingeniería
Universidad de la República

Resumen

Las tecnologías utilizadas por los proveedores de servicios de internet en el despliegue de sus redes internas y en los servicios que ofrecen a sus clientes han ido cambiando con el tiempo como consecuencia de sus crecientes necesidades como ser el bajar los costos de implementación de la red y el bajar los costos de los servicios que ofrecen. También se ha buscado tener mayor facilidad y flexibilidad en el momento del despliegue de las redes, incrementar los tipos de servicios que pueden proveer a sus clientes y minimizar los tiempos de administración de la red, entre otros motivos. Es así que emerge Multiprotocol Label Switching como tecnología predominante para la provisión de servicios de red en general y particularmente de VPNs, y se mantuvo como la tecnología de preferencia por varios años.

Hoy en día nuevas tecnologías compiten en el mercado con MPLS y han ido ganando popularidad, pero la facilidad de despliegue, bajo costo y diversidad de servicios que permite proveer MPLS ha logrado que ésta no quedara obsoleta y siga vigente.

El presente trabajo tiene como objetivo realizar pruebas de escala en la implementación de VPNs de capa 2 y capa 3 basándonos en la tecnología legada MPLS. Los datos obtenidos serán analizados para evaluar su capacidad de escala y serán utilizados en comparación con los de otras tecnologías, como podría ser SDN, para las cuales se haya hecho algún trabajo de éste estilo, de manera de analizar la capacidad de escala de cada una de ellas.

Keywords: Virtual Private Network (VPN), Multiprotocol Label Switching (MPLS), Virtual Private LAN Service (VPLS), Virtual Routing and Forwarding (VRF), Graphical Network Simulator 3 (GNS3).

Tabla de contenidos

1. Introducción y motivación

MPLS se ha establecido por años como la tecnología estándar para el ofrecimiento de servicios de red entre los proveedores de redes privadas e Internet ya sea por su bajo costo de despliegue o por soportar muchos tipos de protocolos de transporte (incluido Ethernet), entre otros muchos beneficios.

Esta tecnología tiene como gran ventaja frente a tecnologías predecesoras el tener separados los planos de control y de datos, pero eso trae como consecuencia el que ambos planos son manejados por cada router nodo que participa de la red MPLS, lo que hace que haya sobrecarga de procesamiento en cada uno de ellos, ya que no solamente se encargan del enrutamiento de los paquetes (PUSH, SWITH y POP de etiquetas) sino que se encargan del establecimiento y mantenimiento de las rutas.

Es por esa razón que se han llevado a cabo investigaciones y trabajos al respecto que han dado como resultado nuevas tecnologías en las que el plano de control pasa a ser manejado por un controlador único en la red que se encarga de calcular las mejores rutas, de establecerlas entre los routers y de mantener toda la información pertinente a su manejo y actualización. De esa manera la carga de trabajo para la realización de esas tareas se libera de los routers nodo de la red y éstos pasan a encargarse solamente del plano de datos.

Actualmente diferentes proveedores utilizan tecnologías basadas en uno u otro acercamiento (incluso algunos proveedores utilizan ambas tecnologías para brindar diferentes servicios) y si bien tecnologías como SDN han ganado bastante terreno, MPLS no ha sido dejado de lado y no está obsoleto, actualmente sigue siendo considerado y utilizado por muchos proveedores para brindar sus servicios. Por ejemplo se utiliza VPN MPLS en entornos de virtualización de datacenters como OpenStack [1] y también OpenDaylight [2] por nombrar algunos.

En el presente trabajo se realiza un estudio sobre la escalabilidad en la configuración de redes privadas virtuales (VPNs por su sigla en inglés: Virtual Private Network) basadas en la tecnología MPLS. Para ello se implementaron herramientas de creación y eliminación automática de VPNs, y se ejecutaron pruebas de escala para poder observar el comportamiento del tiempo de demora en la creación y eliminación de las VPNs, así como del consumo de memoria luego de la creación de las mismas.

El documento actual constituye la documentación final del trabajo, en el marco de la asignatura "Tesis de Licenciatura en Computación" correspondiente al plan de estudios de la carrera de grado "Licenciatura en Computación" de la Facultad de Ingeniería de la Universidad de la República (UdelaR).

Para las pruebas de escala realizadas se implementaron VPNs de capa 2 y 3. Las VPNs de capa 2 se configuraron utilizando VPLS, y las de capa 3 utilizando VRFs. Todas las pruebas fueron realizadas en un entorno emulado con las herramientas GNS3 [3] y Qemu [4].

La realización de este trabajo tiene como motivación la recolección de datos para poder evaluar la escalabilidad de las tecnologías de VPNs basadas en MPLS. Esos datos podrán además ser utilizados en una posterior comparación

con los resultados recabados utilizando otras tecnologías más modernas para la creación de VPNs, como podría ser Redes Definidas por Software (SDN por su sigla en inglés: Software Defined Networking), para las cuales se haya hecho algún trabajo de éste estilo.

2. Trabajo relacionado

En el artículo "VPN scalability over high performance backbones Evaluating MPLS VPN against traditional approaches" [5] Francesco Palmieri realizó un trabajo donde compara la escalabilidad de VPNs MPLS con las tecnologías utilizadas anteriormente (como ser VPNs basadas en túneles) y llega a la conclusión de que las VPNs MPLS escalan mejor que sus predecesoras. Nos proponemos en este trabajo ampliar esas conclusiones realizando pruebas en ambientes de mayor envergadura de manera de obtener datos que reflejen los requerimientos de infraestructura de los proveedores de hoy en día y que nos permitan posicionar la tecnología entre otras que proveen servicios similares.

Se realizó además con anterioridad en la Facultad de Ingeniería de la Universidad de la República un estudio similar analizando la escalabilidad de la tecnología SDN para la configuración de VPNs. Se puede consultar en [6]. Si bien los resultados fueron obtenidos utilizando un entorno diferente al del trabajo actual se podría llegar a obtener una métrica común para poder compararlos.

3. Marco teórico

Se presenta en ésta sección una breve reseña sobre las tecnologías y terminologías utilizadas en el trabajo.

3.1. Redes Privadas Virtuales (VPNs)

Una red privada virtual es una tecnología de redes que permite crear conexiones de redes seguras sobre redes que no lo son, como podría ser Internet o una red de un proveedor de servicios. Es utilizada generalmente para interconectar diferentes sitios de una organización a través de la infraestructura de un proveedor de servicios o Internet, ya que es mucho mas conveniente que utilizar líneas de conexión dedicadas.

Es necesario dar una descripción sobre los diferentes dispositivos de red de los clientes y proveedores que participan en las VPNs:

- Dispositivos del lado del cliente (C): Son dispositivos como routers o switches que residen en la red del cliente y que no están al tanto de las VPNs
- Dispositivos de borde del cliente (CE): Son dispositivos que se encuentran en el borde de la red del cliente y que son conectados a dispositivos del

proveedor (dispositivos PE). Se distinguen entre CE-s (switches) y CE-r (routers). En nuestro laboratorio usamos solamente routers, por lo que utilizamos indistintamente los términos CE y CE-r.
Las VPNs que configuraremos serán basadas en los routers PE por lo que los routers CE tampoco estarán al tanto de las VPNs.

- Dispositivos de borde del proveedor (PE): Son dispositivos que se encuentran en el borde de la red del proveedor y que son conectados a dispositivos CE del cliente. Se distinguen entre PE-s (switches), PE-r (routers) y PE-rs (dispositivos capaces de actuar como routers y switches). En nuestro laboratorio usamos solamente routers, por lo que utilizamos indistintamente los términos PE y PE-r.
  Las VPNs que configuraremos serán basadas en éstos routers por lo que si estarán al tanto de las VPNs.

- Dispositivos del lado del proveedor (P): Son dispositivos como routers o switches que residen en la red del proveedor y que tampoco están al tanto de las VPNs.

Para poder establecer redes privadas virtuales se utilizan diferentes tipos de tecnologías y protocolos. En el presente trabajo nos centraremos en las VPNs de capa 2 y de capa 3 que utilizan Multiprotocol Label Switching (MPLS) como base para su funcionamiento.

MPLS tiene además como gran ventaja que provee un plano de enrutamiento y uno de control totalmente desacoplados, cosa que no sucede con el enrutamiento IP convencional[7].

3.1.1. Modelos de VPN

En los tiempos en que surgió MPLS[1] los proveedores de servicios ofrecían mayormente dos modelos de VPN a sus clientes: el modelo *"overlay"* y el modelo *"peer-to-peer"*.

En el modelo *"overlay"* los routers o switches del proveedor transportan los paquetes de los clientes a través de circuitos o enlaces virtuales utilizando tecnologías de capa 1 o capa 2 como pueden ser TDM en capa 1 y ATM o FrameRelay en capa 2, de manera que los routers del proveedor que trabajan en la capa 3 nunca llegan a ver las rutas de los clientes. Por ejemplo se podría establecer una VPN entre clientes utilizando switches FrameRelay en el proveedor como en la siguiente figura:

[1] MPLS fue especificado en el RFC 3031 publicado en Enero del 2001 [8].

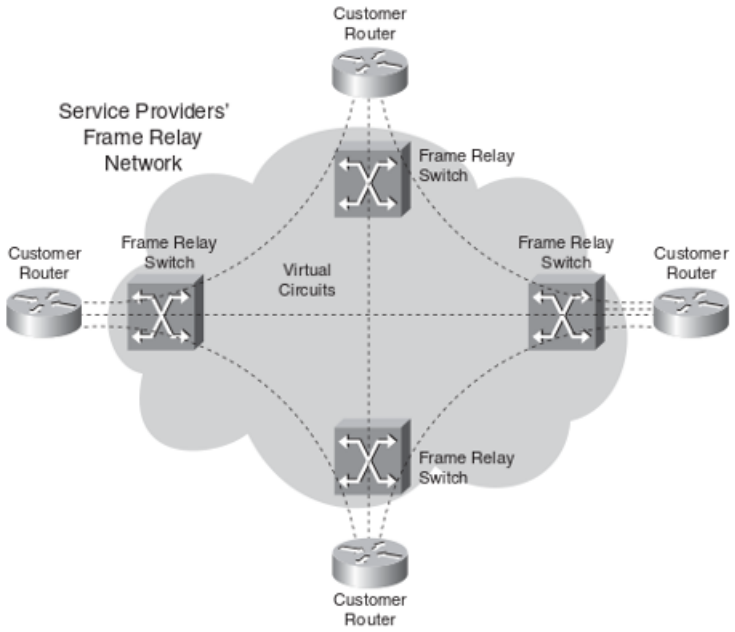

Fig. 1: Red Overlay con Frame Relay[9]

En el modelo *"peer-to-peer"* los routers del proveedor que trabajan en capa 3 son los encargados de transportar los paquetes del cliente, por lo que existe una comunicación entre los routers del cliente y del proveedor utilizando protocolos de enrutamiento. La privacidad y separación de tráfico que requiere la VPN es lograda utilizando filtros de rutas y filtros de paquetes en los routers PE y P del proveedor.

Antes de que existiera MPLS el modelo más común era el *"overlay"* ya que el *"peer-to-peer"* requería mucho más trabajo al agregar sitios de clientes a las VPNs existentes debido a que había que reconfigurar muchos puntos del circuito virtual en forma manual. Las VPNs MPLS son una aplicación de MPLS que ha hecho que el modelo *"peer-to-peer"* fuese mucho más fácil de implementar[9].

La implementación mas madura del paradigma de VPNs MPLS de capa 3 se encuentra descripto en [10] donde se habilita el soporte para VPNs vía MPLS y una extensión multiprotocolo del protocolo BGP (MP-BGP). El protocolo MPLS se utiliza en el enrutamiento de los paquetes mientras que el protocolo BGP se utiliza para distribuir las rutas VPN dentro de la red del proveedor, así como entre diferentes AS de diferentes proveedores.

3.1.2. Multiprotocol Label Switching (MPLS)

MPLS es una tecnología de red que utiliza etiquetas adjuntas a los paquetes para su enrutamiento a través de la red.

En el enrutamiento IP convencional, cuando un paquete atraviesa la red, su encabezado IP es extraído y examinado por cada router, y cada uno de ellos lo asigna a una clase de equivalencia de enrutamiento[2] (FEC por su siglas en inglés: Forwarding Equivalence Class) para luego decidir, en base a sus tablas de enrutamiento, hacia donde dirigir el paquete. Este modo de funcionamiento es conocido como *ip-switching* y es cómo funcionan la mayoría de las redes, incluso Internet.

MPLS permite basar las decisiones de enrutamiento en cada router en base a una etiqueta que se le adjunta al paquete. Este modo de funcionamiento es conocido como *label-switching*[9].

Cuando un proveedor tiene configurado MPLS en su red interna y un cliente le envía un paquete, se le asigna una etiqueta y se le asocia en ese momento a una FEC, y en cada router del camino que atraviesa la decisión de enrutamiento será tomada en base a esa etiqueta (que es de largo fijo y funciona como índice en una tabla de enrutamiento por etiquetas), por lo que el procesamiento se vuelve mucho más rápido y menos costoso en términos de recursos de hardware. En cada router la etiqueta es extraída y sustituida por otra de iguales características. Cuando el paquete abandona la red del proveedor, esa etiqueta es extraída y el paquete original es enviado al destino[8].

MPLS ofrece grandes beneficios frente al enrutamiento IP convencional. Por ejemplo, ya que todo el enrutamiento se basa en etiquetas de largo fijo y en base a una tabla de etiquetas, el enrutamiento MPLS puede ser hecho por dispositivos que manejan solamente hasta capa 2 (como switches) pero que tengan la capacidad de hacer búsquedas y reemplazo de etiquetas. Además se abstrae totalmente de los encabezados de la capa de red (capa 3) por lo que el procesamiento de paquetes es mucho más rápido.

Utilizando MPLS se pueden etiquetar los paquetes para que tomen rutas predeterminadas y lograr separar el tráfico, sea basado en su tipo o en el cliente que lo origine, pudiéndose así establecer Redes Privadas Virtuales (VPNs) entre otros usos. De hecho las VPN sobre MPLS corresponden a la implementación más utilizada de MPLS.

Más información sobre MPLS en general y en particular sobre su arquitectura se puede consultar en *"MPLS Fundamentals"* de CiscoPress[9].

3.1.3. VPNs MPLS de capa 2

Para configurar VPNs de capa 2 sobre MPLS utilizamos VPLS, que es un tipo de red privada virtual de capa 2, point-to-multipoint, basada en Ethernet. Un cliente cuyos sitios son interconectados implementando VPLS tiene la percepción de que sus sitios pertenecieran a la misma LAN.

En VPLS, cuando un paquete llega a un dispositivo PE del proveedor desde uno CE del cliente, éste es etiquetado con MPLS y enviado a través de la red del proveedor por una ruta conocida por su nombre en inglés cómo MPLS Label-Switched Path (LSP).

[2] Una FEC es un grupo de paquetes IP que serán enviados por la misma ruta ya que son tratados de la misma manera en cada router del camino que atraviesan.

Las rutas MPLS LSP que transportan tráfico VPLS entre los routers PE son llamadas pseudowires y son configurados de manera estática o distribuidas con BGP o LDP. En el laboratorio vamos a utilizar el primer tipo (pseudowires estáticos), configurados con la directiva "set protocols mpls label-switched-path" de los routers Juniper[11].

3.1.4. VPNs MPLS de capa 3

Para configurar VPNs de capa 3 sobre MPLS utilizamos el concepto de *"Virtual Routing and Forwarding"* (VRFs), que son tablas de enrutamiento IP virtuales, una por cada VPN, que mantiene el router PE separadas de su tabla de enrutamiento global.

Gracias a las VRFs el proveedor logra la privacidad que requieren las VPN al mantener separada la información de enrutamiento de cada cliente[9].

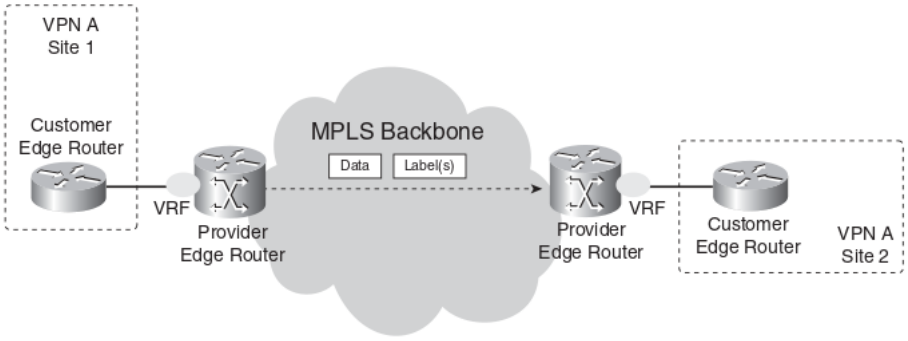

Fig. 2: MPLS VPN con VRFs [9]

Las VRF contienen el mismo tipo de información que las tablas de enrutamiento IP comunes, la diferencia es que son accedidas solamente en caso de que los paquetes sean originados por clientes que pertenezcan a la VPN asociada a la VRF. Cuando un paquete llega a un router PE del proveedor desde un router CE de un cliente que pertenezca a una VPN, es ruteado utilizando la VRF asociada a esa VPN[12].

3.2. Requisitos para la implementación de VPNs con MPLS

Para poder utilizar MPLS en la implementación de nuestras VPNs, en los routers PE y P de la red del proveedor deben estar previamente configurados los siguientes protocolos:

- Un protocolo IGP que puede ser OSPF, RIP o ISIS: Utilizado para lograr conectividad entre los routers.
  En el laboratorio configuramos OSPF en todas las interfaces de los routers P y en las interfaces interiores de los routers PE.

- Label Distribution Protocol (LDP): Es el protocolo utilizado entre los routers para intercambiar información referente a las asociaciones de etiquetas MPLS.
  En el laboratorio lo configuramos en las interfaces interiores de los routers PE.

4. Diseño e implementación del laboratorio

Cómo explican Santiago Vidal, Jorge Rodrigo Amaro, Emiliano Viotti, Martin Giachino y Eduardo Grampin en el trabajo antes mencionado [6] "*A fair quantitative comparison against legacy MPLS is difficult to execute, since open source MPLS implementations, which can be emulated under the same environment as RAUflow, are far less efficient than industrial ones.*".

Uno de los objetivos que nos planteamos en nuestro trabajo es obtener datos que puedan ser utilizados en una futura comparación con los datos allí obtenidos, por eso buscamos una plataforma que nos permitiera utilizar emuladores de routers industriales.

4.1. Plataforma de emulación de la topología y routers

La plataforma utilizada para las pruebas es una red emulada con GNS3, donde se modelaron routers de borde del lado del cliente (routers CE), routers de borde del lado del proveedor de servicios de Internet (routers PE), y routers centrales del núcleo de la red del proveedor de Internet (routers P).

Originalmente se utilizaron routers Cisco 7200 [13][14] para emular la topología de red ya que requerían menos recursos de hardware y Cisco es uno de los proveedores más utilizadas en Internet, pero las características de la tecnología necesaria para implementar VPLS no están disponibles en sus entornos emulados, así que tuvimos que comenzar a utilizar routers Juniper [15], que en su versión vSRX si lo soporta [16].

Cómo al iniciar el trabajo se utilizaban routers Cisco, se comenzó utilizando GNS3 debido a que éste maneja en forma nativa dynamips, el emulador existente mas liviano para Cisco. Además es una herramienta muy flexible y permite en muy poco tiempo tener funcionando cualquier tipo de escenario. Es cierto que una vez que comenzamos a utilizar routers Juniper en lugar de Cisco pudimos haber comenzado a utilizar otra herramienta que no sea GNS3 (más que nada por tener la desventaja de necesitar una interfaz gráfica y no permitir ser manejado por línea de comandos), pero nos mantuvimos con GNS3 por ser una herramienta que muy fácilmente nos permitió generar un laboratorio para nuestras pruebas.

Actualmente todos los routers emulados son Juniper y utilizan JunOS, el sistema operativo común a todos los dispositivos Juniper. Los routers CE y P utilizan la versión Olive (que es una implementación de JunOS preparada para correr en equipos x86 en lugar de hardware de routers) mientras que los routers

PE utilizan vSRX (que es una implementación de JunOS que funciona como un firewall virtualizado preparado para correr sobre hardware de virtualización) ya que las características necesarias para configurar las VPN de capa 2 (VPLS) en los routers de borde del lado del proveedor solo eran provistas por esa versión para los entornos emulados[3]. Más adelante explicaremos con más detalle ésta limitación.

Se utilizan esas dos versiones diferentes de routers Juniper por un tema de recursos, ya que los vSRX necesitan el doble de recursos que los Olive (2048MB los routers vSRX vs. 1024MB los routers Olive).

La configuración de las VPNs con Juniper tanto de capa 2 como de capa 3 se basan en el concepto de "routing instance" que es una colección de tablas de ruteo, interfaces y parámetros de los protocolos de ruteo [17]. Hay 12 tipos de routing instances definidos por Juniper. Para el caso de VPNS de capa 2 utilizamos routing instances del tipo VPLS y para el caso de VPNs de capa 3 utilizamos routing instances del tipo VRF. Por cada VPN que configuremos se ingresa una routing instance en los routers PE involucrados.

4.2. Requisitos de la topología

Lograr un buen diseño de una topología de red es un problema que no tiene una solución óptima única. Lo que se obtiene luego de un proceso cuidadoso de diseño es un conjunto de soluciones posibles. Una muy buena discusión sobre ésta problemática se desarrolla en el artículo "IGen: Generation of Router-level Internet Topologies through Network Design Heuristics" de Bruno Quoitin, Virginie Van den Schrieck, Pierre Francois y Olivier Bonaventure [18]. Allí se discuten los objetivos que se buscan al diseñar una topología de red. De allí hemos recogido tres lineamientos fundamentales en los que nos basaremos para generar una topología que nos brinde datos realistas. Esos lineamientos son:

1. minimizar la latencia,
2. agregar redundancia de manera que se pueda optar por una ruta alternativa en caso de que deje de estar en línea la ruta más corta, y
3. minimizar los costos.

Claramente los lineamientos 1 y 2 contrastan con la minimización de los costos del punto 3 por lo que hicimos un balance entre cantidad de equipos y enlaces, además hay otro requisito que se desprende del objetivo que nos propusimos de crear una topología de red que posea una cantidad determinada de routers de manera de emular una red de mediano porte. Teniendo en cuenta estos lineamientos, diseñamos una red con 62 routers: 36 CE, 12 PE y 14 P, y los agrupamos de la siguiente manera:

[3] Cabe destacar que la versión Olive de JunOS se puede usar libremente siempre que sea con propósitos de estudio y que la versión vSRX si bien es comercial puede descargarse como un trial de 60 días en https://www.juniper.net/us/en/dm/free-vsrx-trial/ cómo se hizo en éste caso.

- Cada sitio de un cliente esta representado por 6 routers CE adyacentes
- El proveedor se conecta a los diferentes sitios de los clientes a través de 2 routers PE que configuran un Point-of-presence (POP)[4]
- Cada router PE esta conectado a tres routers CE y a dos routers P. Además son agrupados de a dos de manera que representen dos puntos de acceso al proveedor en cada cliente.
- Cada router P esta conectado a otros cuatro routers P.

De esa manera creamos una red que simula un proveedor de mediano porte y nos aseguramos que la distancia máxima entre routers PE sea de 6 saltos minimizando la latencia. Además logramos que haya redundancia en todos los enlaces entre los routers PE y los P, y entre los diferentes routers P también. Tal redundancia no es necesaria realmente porque los enlaces en sí van a estar siempre en línea, pero le damos más realismo al laboratorio ya que los protocolos de enrutamiento tienen que evaluar y tomar decisiones respecto a los diferentes caminos para alcanzar una misma red. Entre los routers CE_n y PE_n se creó solo un enlace por simplicidad ya que tener redundancia allí no aporta en nuestro trabajo.

Para la generación automática de la topología del laboratorio se provee un script: "topology_maker.sh" que sigue los lineamientos anteriormente descriptos. Sus parámetros de ejecución son los siguientes:

- project_name: es el nombre que tendrá el proyecto GNS3
- total_number_of_routers: es el número total de routers que va a tener la topología
- number_of_ce_routers: es el número de routers CE

Esta herramienta genera una carpeta con el proyecto GNS3 "project_name" con un total de "total_number_of_routers" routers, de los cuales "number_of_ce_routers" serán routers CE. Habrá "number_of_ce_routers/3" cantidad de routers PE. En esa carpeta se encuentra el archivo .gns3 correspondiente al proyecto y es el archivo que habrá que abrir con GNS3. Tal archivo utiliza el formato de la versión 1.2.3 de GNS3, pero puede ser abierto con cualquier versión posterior.

IMPORTANTE: Es necesario que luego de la creación del proyecto se edite manualmente tal archivo para reemplazar en cada caso necesario la ruta del ejecutable del emulador Qemu y la ruta de las imágenes .img de los routers.

Los routers CE y P se generan utilizando la imagen compatible con Qemu "Junos-Olive.img". Tal imagen fue creada de cero, instalando primero FreeBSD 4.11 desde la imagen ISO "4.11-RELEASE-i386-mininst.iso" y luego instalando el paquete "jinstall-10.1R1.8-domestic-olive.tgz"[19]. A tales routers se les asigna 1GB de memoria RAM.

[4] Un Point-of-presence (POP) es un punto de acceso de un cliente a Internet a través de un proveedor.

Los routers PE se generan utilizando la imagen "Junos-vSRX.img" y se les asignó 2GB de memoria RAM. Tal imagen fue modificada para que los routers no funcionen como firewall, cambiando el modo del forwarding de paquetes de "flow based" a "packet based" con los siguientes comandos del Command Line Interface (CLI) de JunOS:

1. delete security
2. set security forwarding-options family mpls mode packet-based
3. set security forwarding-options family iso mode packet-based
4. set security forwarding-options family inet6 mode packet-based
5. commit

IMPORTANTE: Luego de que se termine de ejecutar el "commit" hay que reiniciar el router para que tome los cambios. Es necesario notar que en las posteriores ejecuciones del "commit" (incluso luego del primer reinicio), se mostrará una advertencia diciendo que se cambió el modo de forwarding y que es necesario reiniciar el router, pero tal advertencia se debe ignorar si es que ya ha sido reiniciado al menos una vez luego del cambio.

Con respecto a porqué se utiliza vSRX y no Olive, en las versiones emuladas previas a vSRX (como Olive o vMX) las interfaces de red del tipo e1000 agregadas en Qemu eran configuradas por JunOS como interfaces "em-*" y cuando se intenta configurar una VPN de capa 2 con VPLS aparece un error que informa que dichas interfaces no soportan VPLS. Por otro lado JunOS en los routers vSRX ve las interfaces de red del tipo e1000 como "ge-*", y tales interfaces sí soportan VPNs de capa 2 con VPLS. Por esta misma razón es que utilizamos interfaces e1000 en Qemu, ya que son interfaces Ethernet Gigabit que luego serán vistas como "ge-*" por JunOS en los routers vSRX.

Los routers deben tener al menos 6 interfaces cada uno y deben ser del tipo e1000 para su correcta emulación con Qemu. Éste requerimiento es debido a algunas limitaciones y errores de versiones antiguas de GNS3 (anteriores a la versión 1.2.3) que se corregían teniendo al menos esa cantidad de interfaces de red por dispositivo.

Para poder automatizar la configuración de las direcciones IP de la red, la creación y configuración de VPNs y la toma de tiempos, es necesario conocer de antemano que routers CE, PE y P están interconectados y a través de que interfaces. Es por eso que se deben proporcionar tres archivos CSV. El primero debe llamarse "topology_description.txt" y debe residir en la misma carpeta que los scripts. Allí se deben indicar los enlaces entre los routers CE y los PE. Por ejemplo en ese archivos se indica que el router CE_i está conectado a través de la interfase "em$k$" a la interfase "ge-0/0/m" del router PE_j.

Se debe especificar en cada línea los siguientes valores separados por una coma (","):

- Identificador del router CE (número i)

- Identificador del router PE (número j)
- Índice de la interfase del router CE_i (número k)
- Índice de la interfase del router PE_j (número m)

Los otros dos archivos deben llamarse "topology_description_pe.txt" y "topology_description_p.txt", y deben residir en la misma carpeta que los scripts. El contenido de ambos archivos es similar a "topology_description.txt". El primero enumera los enlaces entre los routers PE y los P, y el segundo los enlaces entre los routers P.

La utilización de los archivos de configuración anteriormente descriptos nos dan independencia entre la herramienta utilizada para generar la topología a utilizar y las herramientas creadas en el laboratorio. Es así que se podrían utilizar para generar la topología herramientas como "IGen" (http://igen.sourceforge.net/) o similares, o descargar topologías pre-hechas como en el caso de "The Internet Topology Zoo" (http://www.topology-zoo.org/).

Otro archivo de configuración importante es "router_ports.sh" que contiene las direcciones IP, los puertos de administración de cada router, y la cantidad de routers CE, PE y P que hay. Éste archivo debe residir en la carpeta "routers_managment".

Como ejemplo los archivos utilizados en el laboratorio se muestran en los Apéndices B, C, D y E.

4.3. Topología utilizada en el laboratorio

La topología de red utilizada en el laboratorio fue creada con la herramienta "topology_maker.sh" mencionada en la sección anterior y se muestra en la siguiente figura. Comprende 36 routers CE (en la parte exterior de la figura), 12 routers PE (dentro de las nubes) y 14 routers P (en la parte interior) interconectados como muestra la Fig. 3.

A los routers Olive se les asignó 1GB de memoria RAM mientras que a los vSRX se les asignaron 2GB.

Para su emulación se utilizó Qemu integrado en GNS3, y para su administración y configuración se utilizaron los puertos seriales que brinda Qemu para cada máquina virtual. Se utiliza TELNET en dichas conexiones.

El equipo en el que corren el GNS3 y las máquinas virtuales del laboratorio cuenta con 128GB de memoria RAM y 32 CPUs. El laboratorio emulado con GNS3 utiliza 74GB[5].

4.4. Herramientas implementadas

Para el laboratorio se implementaron un conjunto de herramientas para automatizar todos los procesos de configuración de la red, configuración y borrado

[5] (36+14)*1 + 12*2 = 74

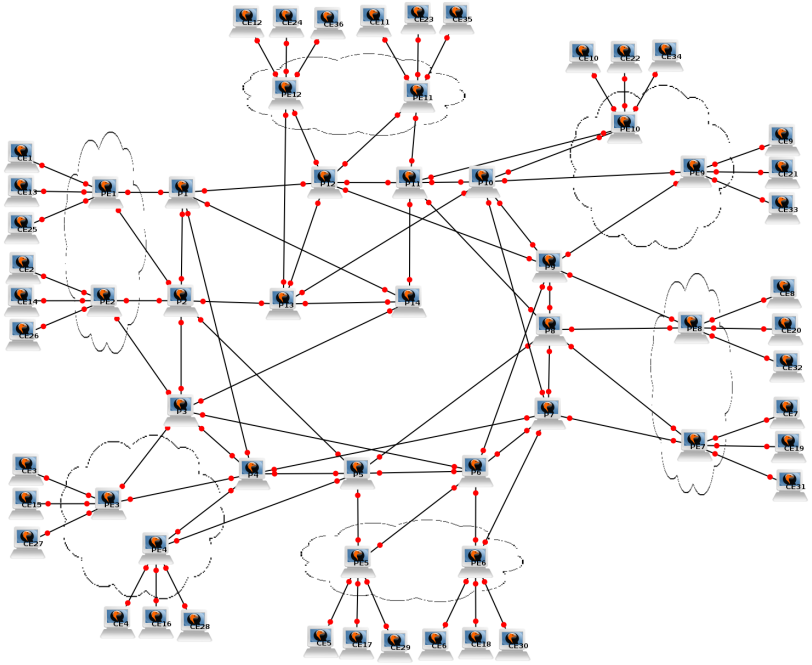

Fig. 3: Topología del laboratorio. Imagen generada con GNS3

de las VPNs, y medición de los tiempos correspondientes. Para su correcta ejecución es necesario que haya sido creada una topología de red que cumpla los requisitos de la sección anterior.

Cuando se utiliza Qemu con GNS3 se habilitan dos puertos seriales de comunicación para conectarse con cada router : uno para la consola de administración del router y otro para la consola de administración de la instancia del Qemu que corre el router (llamado monitor). En el laboratorio vamos a utilizar la consola de administración de los routers para setear las VPN. Nos vamos a comunicar a través del puerto serial utilizando TELNET y automatizaremos las tareas con scripts de EXPECT.

Se proporciona una herramienta ("get_ports_from_gns3_file.sh") para el caso de no haber creado manualmente el archivo de configuración "router_ports.sh" descripto en la sección anterior. Se debe ejecutar pasándole como parámetro el nombre del proyecto GNS3 que vayamos a utilizar. Esto genera el archivo "router_ports.sh" que contiene:

- las direcciones IP y puertos de administración de cada router,
- la cantidad de routers CE, PE y P que hay en la topología.

Una vez que tengamos creados los archivos de configuración necesarios y hayamos generado el proyecto, lo abrimos con GNS3 e iniciamos los routers.

Todas las herramientas suministradas en el trabajo requieren que los routers se encuentren en la Command Line Interface (CLI). Para automatizar el ingreso de todos los routers ejecutar el script "enter_cli_routers.sh" que ingresa en la CLI de cada router y los deja prontos para recibir comandos.

Para que todo funcione correctamente, las interfaces de los routers deben estar todas en estado "Admin Up". Hay veces que al iniciar los routers las interfaces de los CE están deshabilitadas[6] o las de los PE no hay sido reconocidas correctamente como interfaces "ge-*", por lo que se recomienda ejecutar la herramienta "check_interfaces.sh". En caso de que el resultado de alguna interfase de un router CE o P sea negativo ejecutar la herramienta "rise_up_interfaces_ce_routers.sh" pasándole la ID del router como parámetro[7], y en caso de que el resultado de alguna interfase de un router PE sea negativa reiniciar dicho router[8].

Una vez que todos los routers se encuentran en la CLI y las interfaces están habilitadas, debemos ejecutar la herramienta "configure_network.sh" que asigna direcciones IP a todas las interfaces del laboratorio y genera las interfaces

[6] Esto ocurre generalmente en caso de haber reiniciado el router ya que JunOS al reiniciar deshabilita las interfaces que están configuradas como "vlan-tagging".

[7] En caso de ser múltiples routers los que necesiten corrección se puede ejecutar la herramienta "rise_up_interfaces_ce_routers_complete.sh" que no lleva parámetros y rehabilita las interfaces de todos los routers CE.

[8] Hay disponibles múltiples herramientas para el reinicio masivo de los routers, todas comienzan con "restart_". En caso de necesitar reiniciar los routers CE ejecutar "restart_ce_routers.sh", en caso de necesitar reiniciar los routers PE ejecutar "restart_pe_routers.sh". También se puede ejecutar la herramienta "restart_router.sh" que reinicia un solo router, cuyo nombre debe ser suministrado como parámetro (por ejemplo "CE15").

loopback en los routers PE necesarias para el correcto funcionamiento del protocolo BGP. Si todo funciona correctamente, al finalizar su ejecución y luego de algunos minutos cada router PE debería ser capaz de comunicarse con las interfaces loopback de todos los otros routers PE.

Cumplido éste paso el laboratorio queda pronto para realizar las pruebas de creación y borrado de VPNs capa 2 y 3.

4.5. Creación y borrado de las VPNs

La herramienta que se utilizará para la creación y borrado, así como para la medición de los tiempos es "complete_cycle.sh", la cual genera automáticamente las VPNs y las configura en los routers CE y PE.
Sus parámetros de ejecución son los siguientes:

1. nr_cycles: la cantidad de veces que se ejecutará la medición de los tiempos de creación y borrado de las VPNs.
2. cant_vpns: es la cantidad de VPNs que se configurarán en la topología.
3. max_cant_of_ces_in_vpns: es la cantidad de routers CE que como máximo participarán en cada VPN.
4. layer: es la capa sobre la que se configurarán las VPNs.
5. add_to_last_run: establece si la actual medición debe ser hecha a partir de alguna medición anterior (permite medir tiempos incrementales). Sus posibles valores son 0 y 1. En caso de establecerse en 1 debe indicarse además: id_last_run: es el ID de la ejecución a la que se le quiere agregar las VPNs generadas. Las mediciones se van numerando de 1 en adelante, siendo 1 la medición cuya ejecución tuvo un 0 en el parámetro "add_to_last_run".

La ejecución se realiza 2 pasos:

- Un paso inicial donde se configuran las VPNs ingresando cada comando en las CLI de los routers utilizando TELNET y luego salvando la configuración con una ID específica.
- Un paso final donde iterativamente se configuran las VPNs cargando la configuración salvada en la primera etapa y midiendo los tiempos.

Se realiza de ésta manera ya que configurar manualmente las VPNs ingresando cada comando en las CLI con TELNET toma mucho más tiempo que cargar la configuración de un archivo previamente guardado. Como el fin del presente trabajo es realizar pruebas midiendo los tiempos de creación y propagación de las VPN por los routers luego de configuradas, el tiempo del paso inicial será descartado.

A su ves cada paso se divide en dos etapas:

- Paso inicial:

- etapa de configuración de las VPNs:
  - entrar en el modo de configuración
  - configuración de las VPNs
  - guardado de la configuración
  - commit
  - salir del modo de configuración
  - espera de la activación de las VPNs
- etapa de eliminación de las VPNs:
  - borrado de las VPNs
  - commit
  - salir del modo de configuración
  - espera de la eliminación de las VPNs

- Paso final (iterativo):
  - etapa de configuración de las VPNs:
    - entrar en el modo de configuración
    - carga de la configuración anteriormente guardada
    - commit
    - salir del modo de configuración
    - espera de la activación de las VPNs
  - etapa de eliminación de las VPNs:
    - borrado de las VPNs
    - commit
    - salir del modo de configuración
    - espera de la eliminación de las VPNs

En caso de que el argumento 4 al convocar la herramienta sea "1", en la primera etapa se carga la configuración cuyo ID es el 5to argumento: "id_last_run". Ésta opción nos permitirá crear incrementalmente VPNs para poder responder preguntas como: ¿cuanto tiempo demora en configurarse una VPN luego de que en el router ya hay configuradas otras X?

Se proporciona además otra herramienta para retomar las mediciones en caso de que la ejecución actual falle[9]: "complete_cycle_fixer.sh".
Tiene como parámetros:

1. end_cycle: la cantidad final de ciclos de esa corrida que falló

[9] Han habido casos en que algún router deja de responder, quizá por sobrecarga del equipo que está corriendo las máquinas virtuales. En tal caso se debe obtener el puerto de monitorización de la máquina virtual que falló (ps -elf | grep CExx y buscar la sección "monitor"), se debe acceder con TELNET a ese puerto y se debe ejecutar el comando system_reset para reiniciar el router. Luego de que reinicie se debe ejecutar la herramienta "enter_cli_routers.sh", luego "delete_vpnl3.sh ID/1;./wait_for_routers.sh" y luego "rise_up_interfaces_ce_routers.sh".

2. init_cycle: el ciclo en que debe retomar la medición
3. id_last_run: el ID de la ejecución que falló
4. layer: es la capa sobre la que están siendo configuradas las VPNs

Para su ejecución es necesario que haya pasado ya por el primer ciclo, de manera que exista en los routers un archivo de configuración de la ejecución ID.

4.6. Medición de los tiempos

Los tiempos medidos en el paso iterativo de cada ciclo de la ejecución son:

1. Tiempo de las sub-etapas correspondientes a la espera de la activación de las VPNs. Es el tiempo que demoran los routers PE en establecer las VPNs y dejarlas activas para la comunicación entre los routers CE correspondientes. Es el tiempo que más nos interesa porque nos muestra el tiempo que demoran los routers PE en establecer las conexiones necesarias para crear las VPNs y nos va servir para sacar conclusiones sobre la escalabilidad de la tecnología que estamos evaluando.
2. Tiempo total de creación de las VPN. Éste tiempo incluye las sub-etapas de ingreso en el CLI, ejecución del comando "load", ejecución del comando "commit", ejecución del comando "exit" y espera de activación de las VPNs.
3. Tiempo de las sub-etapas correspondientes a la espera de la eliminación de las VPNs.
4. Tiempo total de eliminación de las VPN. Éste tiempo incluye las sub-etapas de ingreso en el CLI, ejecución de los comandos de eliminación de las VPN, ejecución del comando "commit", ejecución del comando "exit" y a la espera de la eliminación de las VPNs.

Los tiempos obtenidos en cada ejecución son guardados en la tabla "times" de una base de datos PostgreSQL llamada "tesis" que se ejecuta en el mismo servidor.

Las columnas de la tabla son:

- id: es la ID de la ejecución
- cycle: es el número de ciclo dentro de la ejecución
- c_commit: es el tiempo 1 de la sección anterior
- c_total: es el tiempo 2 de la sección anterior
- d_commit: es el tiempo 3 de la sección anterior
- d_total: es el tiempo 4 de la sección anterior
- cant_vpns: es la cantidad de VPNs configuradas en la ejecución

- nr_run: es el número de medición en la ejecución actual (ver sección 3.5, parámetro 4 de la ejecución de la herramienta "complete_cycle.sh")
- layer: es la capa en la que fueron configuradas las VPNs (2 o 3)

Para cada ejecución ID se van a guardar un número de tuplas igual a la cantidad de ciclos que se le pasaron a la herramienta "complete_cycle.sh" en su ejecución.

4.7. Medición del consumo de memoria en los routers PE

Para analizar la escalabilidad de las VPNs basadas en MPLS, además de medir los tiempos de demora en la creación de las VPNs vamos a medir el consumo de memoria en los routers PE luego de creadas las mismas. Para ello se proporciona una herramienta ("get_memory_usage.sh") que debe ser ejecutada al final de la toma de los tiempos. No requiere parámetros y lo que hace es recorrer las diferentes corridas de los experimentos y tomar nota del promedio del consumo de memoria de los routers en cada caso.

Los resultados son guardados en la tabla "memory_usage" de la base de datos "tesis" descripta en la sub-sección anterior. Las columnas de la tabla son:

- id: es la ID de la ejecución
- cant_vpns: es la cantidad de VPNs configuradas en la ejecución
- layer: es la capa en la que fueron configuradas las VPNs (2 o 3)
- total_memory: es la memoria total disponible actualmente en el router luego de configurar las VPNs
- control_plane_memory: es la memoria del plano de control disponible actualmente en el router luego de configurar las VPNs
- data_plane_memory: es la memoria del plano de datos disponible actualmente en el router luego de configurar las VPNs

Recordemos que la memoria total de los routers inicialmente es 2048 MB de los cuales 1150 MB se reservan para el plano de control y los otros 898 MB se reservan para el plano de los datos. Esta información se obtiene ejecutando el comando "show chassis routing-engine" de la CLI.

5. Pruebas de concepto y análisis de escalabilidad

Se realizaron las mediciones de tiempos configurando de a múltiplos de 100 VPNs en el rango de 100 a 1000.

Se crearon un máximo de 1000 VPNs debido a que existe una limitación en cuanto a la cantidad de VPNs que podemos configurar en cada router ya que para poder configurar diferentes VPNs en una misma interfase en un mismo

router se debe particionar la interfase en sub-interfaces. Esto se logra configurando VLANs entre los routers CE y PE. La limitación se halla en que cada VLAN debe tener una ID (vlan-id) y las diferentes versiones de sistema JunOS que manejamos en el laboratorio (Olive en caso de los routers CE y vSRX en caso de los routers PE) tienen diferentes rangos de IDs permitidas. En la versión Olive el rango es de 1 a 1024 mientras que en los vSRX el rango es de 512 a 4096. El problema es que en cada caso hemos conectado una interfase de un router CE con una de un router PE por lo que solo podemos utilizar las ID que pertenezcan a la intersección de ambos rangos, o sea de 512 a 1024. Cabe destacar además que cada router por cada VLAN para una misma interfase debe tener un vlan-id único.

Para explicar lo que se hizo al respecto en este trabajo, consideremos el paso "n" de la medición de tiempos. El procedimiento para la elección de la vlan-id al configurar la VPNn es el siguiente:

1. se chequea cual es la mayor vlan-id de entre los routers que participarán en dicha VPN,

2. se toma el valor vlan-id+1.

De esa manera nos aseguramos que cada VLAN configurada en cada interfase de cada router tenga una vlan-id diferente y que si dos grupos de routers se configuran en dos VPNs diferentes las interfaces pueden llegar a tener la misma vlan-id sin que se generen conflictos.

Al realizarlo de esa manera podemos configurar más de las 512 VPNs que nos permiten los posibles valores del rango.

De todas maneras, la limitación nos topea a que como máximo un router puede participar en a lo sumo 512 VPNs en nuestro laboratorio, por lo que configurar más de 1000 VPNs podría resultar en que alguno de los routers no aceptara las configuraciones y el test falle, es pos eso que nos limitamos en la creación de 100 VPNs como máximo.

Para obtener una cota superior de la cantidad de VPNs que podríamos configurar antes de que el laboratorio falle podríamos hacer lo siguiente: luego del último paso de las pruebas de capa 2 con la creación de 1000 VPNs observamos que el máximo número de VPNs configuradas fue de 395[10], lo que arroja que como máximo se configuraron 0.395 VPN en cada router CE por cada VPN configurada en el laboratorio[11], por lo que utilizando las herramientas implementadas se podrían configurar un máximo de 117 VPNs más por cada router CE[12], lo que daría un número general de 117/0.395=296 VPNs adicionales a las 1000 que ya hemos configurado.

[10] Obtuvimos el valor promedio ejecutando "*cat *vlan_id.txt | awk '{print $1 - 512}' | sort | tail -1*" en la carpeta creada en la corrida del test con 1000 VPNs de capa 2.
[11] 0.395=(1*395/1000).
[12] 117=512-395.

5.1. Resultados esperados

Lo que esperamos obtener como resultado de los siguientes tests es un incremento del factor tiempo y de la memoria consumida en los routers en relación directa a la cantidad de VPNs configuradas. Este comportamiento se debe a diferentes factores según la capa en la que se esté configurando la VPN.

Con respecto al incremento del tiempo de creación, en capa 2, cuando en un router PE establecemos las VPNs utilizando routing instances del tipo VPLS, en cada una de las routing instances ingresamos la dirección IP de los otros routers PE que participan en la VPN y luego de hacer el commit el router utiliza el protocolo LDP para intercambiar información con los otros routers sobre las etiquetas que van a ser utilizadas para establecer el camino que atravesarán los paquetes pertenecientes a las VPNs configuradas, por lo que cuanto más VPNs configuremos más información tendrán que intercambiar los routers utilizando el protocolo LDP y más tiempo van a demorar en quedar prontas las VPNs. En capa 3 el comportamiento es similar al caso anterior solamente que se utiliza el protocolo MP-BGP para intercambiar información con los otros routers sobre las etiquetas que van a ser utilizadas para establecer el camino que atravesarán los paquetes pertenecientes a las VPNs configuradas. Es por eso que también se debería observar que un aumento en el número de VPNs configuradas implique un aumento lineal de la cantidad de información que tendrán que intercambiar los routers y aumento al menos lineal del tiempo que van a demorar en quedar operacionales las VPNs (al menos, porque también podría haber un aumento en la probabilidad de colisiones y necesidades de retransmisión en los protocolos de las capas inferiores, así como un aumento en los tiempos de espera que pueden tener los paquetes en los buffers de entrada de los routers por estar éstos procesando otros paquetes). Además, está el hecho de que a estos tiempos deben sumarse los tiempos que los routers necesitan para generar secuencialmente los mensajes a enviar al inicio de la prueba, lo que razonablemente debe crecer linealmente con la cantidad de VPNs.

En ambos casos los mensajes que deben intercambiar los routers para establecer las rutas son siempre del mismo tamaño y si bien la cantidad varía según cuantos routers participen en las VPNs, eligiendo un número aleatorio entre 1 y *max_cant_of_ces_in_vpns* (el parámetro en la ejecución de *complete_cycle.sh*) de routers participantes para cada una de las n VPNs, y existiendo por lo tanto una probabilidad equivalente de elegir cualquiera de los números entre 1 y *max_cant_of_ces_in_vpns*, el promedio de routers participantes por VPN debería tender a ser (en números grandes) de $(max\_cant\_of\_ces\_in\_vpns+1)/2$. De esa manera como el tamaño y la cantidad de paquetes que se intercambian para establecer cada VPN se puede tomar como constantes, es de esperar que el crecimiento del factor tiempo con respecto a la cantidad de VPNs configuradas crezca al menos linealmente (por las consideraciones del párrafo anterior).

Una suposición análoga se puede hacer en el caso del tiempo de borrado de las VPNs.

Con respecto al incremento de la memoria consumida en los routers, al ingresar cada VPN se ingresa una nueva routing instance la cual posee entre otras

cosas una tabla de ruteo propia que es mantenida en memoria por el router. Es por eso que se puede esperar que un aumento en el número de VPNs configuradas implique que crezca el espacio de memoria que se necesite. Cabe recordar que los valores obtenidos de la memoria consumida por los routers en cada corrida del test corresponden al promedio del consumo de memoria de los routers. Por ejemplo en el caso de la corrida del test correspondiente a la configuración de 300 VPNs de capa 2, se obtuvo un total de memoria consumida de 596 MB, esto corresponde al total de memoria consumida promedio entre todos los routers PE para ese caso particular: memoria_total = (memoria_router_PE1 + memoria_router_PE2 + ...)/12 . Hay que tener en cuenta que ese promedio es matemático y no considera el hecho de que algunos routers PE participen en más VPNs que otros, lo que podría ocasionar que el promedio baje o suba independientemente del número de VPNs que se estén configurando en ese momento en total. Por ejemplo supongamos que contamos con 4 routers y en la corrida de 200 VPNs dan como resultado 190, 185, 182 y 184 MB de memoria consumida, su promedio sería 185.25. Ahora supongamos que en la corrida de 300 VPNs uno de los routers por tener menos VPNs configuradas tiene menos memoria ocupada por lo que los valores de memoria consumida son 210, 215, 203 y 60 MB lo que da como promedio 172. Por lo tanto en el laboratorio si bien presentamos los valores de memoria consumida en cada corrida, lo que nos va a interesar para evaluar la escalabilidad o no de la tecnología es el valor de la memoria consumida en la corrida correspondiente a la configuración de 1000 VPNs.

5.2. VPNs capa 2

Se corrió el siguiente script para realizar los tests de configuración de VPNs de capa 2:

```
#!/bin/bash

for i in $(seq 1 10)
do
        ./complete_cycle.sh 10 $(( $i*100 )) 5 2 0
done
```

donde los parámetros se establecen en:

- 10: cada test debe ejecutarse “10” veces
- $(($i*100)): se deben configurar “$(($i*100))” VPNs
- 5: cada VPN debe configurarse entre 1 y “5” routers
- 2: deben configurarse VPNs de capa “2”
- 0: se debe ejecutar cada test sin partir de ninguna medición anterior.

Luego de su finalización, para obtener los promedios de los tiempos de ejecución (sin considerar el primer ciclo de cada una de ellas) se ejecutó la consulta:

```
select
        cant_vpns,
        round(avg(c_commit)::numeric,0) as c_commit,
        round(avg(c_total)::numeric,0) as c_total,
        round(avg(d_commit)::numeric,0) as d_commit,
        round(avg(d_total)::numeric,0) as d_total
from
        times t
where
        cycle>1 and
        nr_run=1 and
        layer=2
group by
        cant_vpns
order by
        cant_vpns asc;
```

Se obtuvieron los siguientes resultados:

cant_vpns	c_commit	c_total	d_commit	d_total
100	5	135	6	109
200	22	186	6	117
300	52	228	6	130
400	65	269	6	139
500	87	337	6	149
600	104	339	6	154
700	135	408	6	156
800	161	446	6	164
900	200	557	6	185
1000	225	566	6	188

Tab. 2: Resultados del tests de capa 2

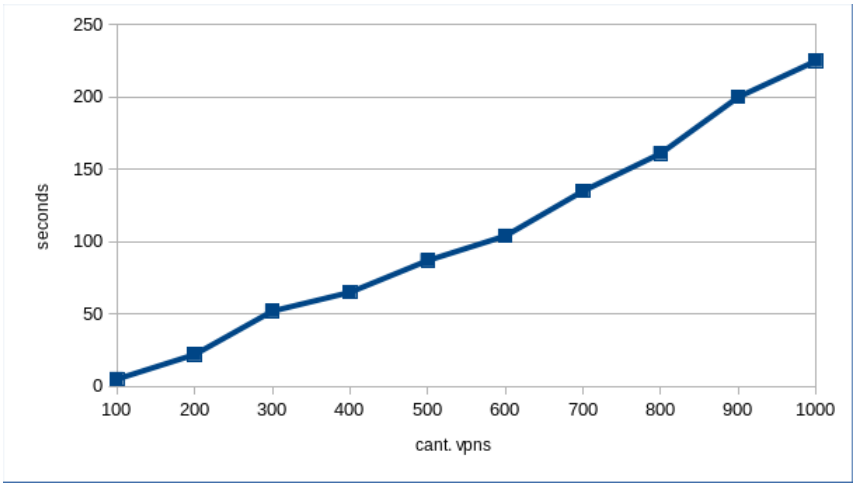

Fig. 4: Tiempo de creación de las VPNs de capa 2. Gráfico generado con Libreoffice Calc

Allí se aprecia claramente que el tiempo que demora en la creación de las VPNs crece casi linealmente a la cantidad de VPNs.

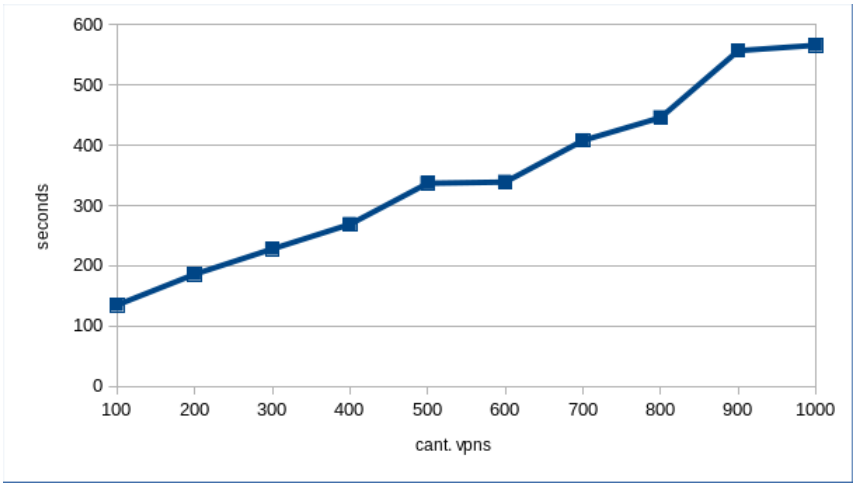

Fig. 5: Tiempo de creación de las VPNs de capa 2 incluyendo la configuración. Gráfico generado con Libreoffice Calc

Lo mismo sucede con el tiempo total, si bien la cantidad de segundos es mayor, el comportamiento es el mismo.

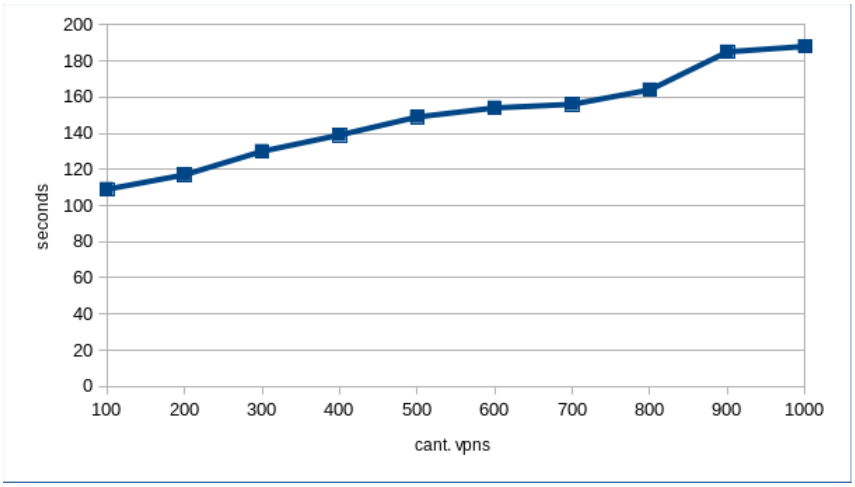

Fig. 6: Tiempo de borrado de las VPNs de capa 2. Gráfico generado con Libreoffice Calc

Aquí se observa que el tiempo de eliminación de las VPNs también crece casi linealmente a la cantidad de VPNs.

Para obtener el consumo de memoria en los routers PE luego de la medición de los tiempos anteriormente descriptos se ejecutó la consulta:

```
select
        cant_vpns,
        round(total_memory) as total,
        round(control_plane_memory) as control_plane
from
        memory_usage
where
        layer=2
order by
        cant_vpns;
```

Se obtuvieron los siguientes resultados:

cant_vpns	total	control _plane
100	594	393
200	594	391
300	596	392
400	596	393
500	594	391
600	594	394
700	599	398
800	602	398
900	614	403
1000	614	403

Tab. 4: Resultado de las mediciones de memoria de capa 2

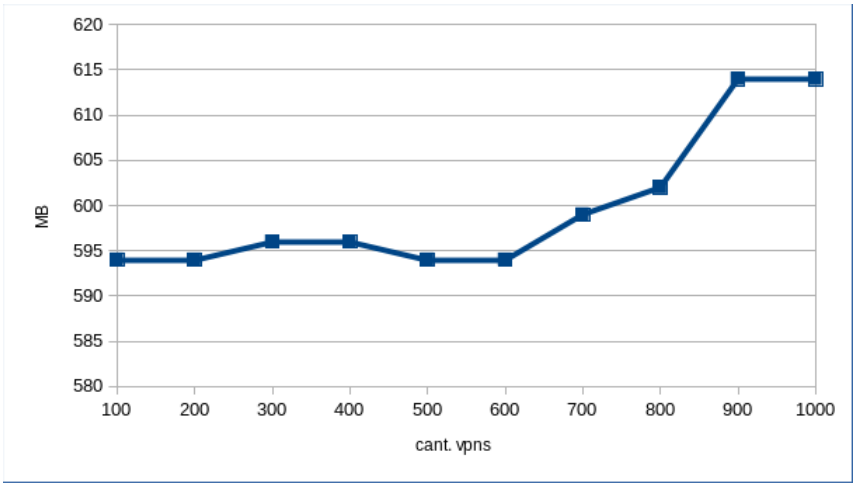

Fig. 7: Memoria total utilizada en los routers PE. Gráfico generado con Libreoffice Calc

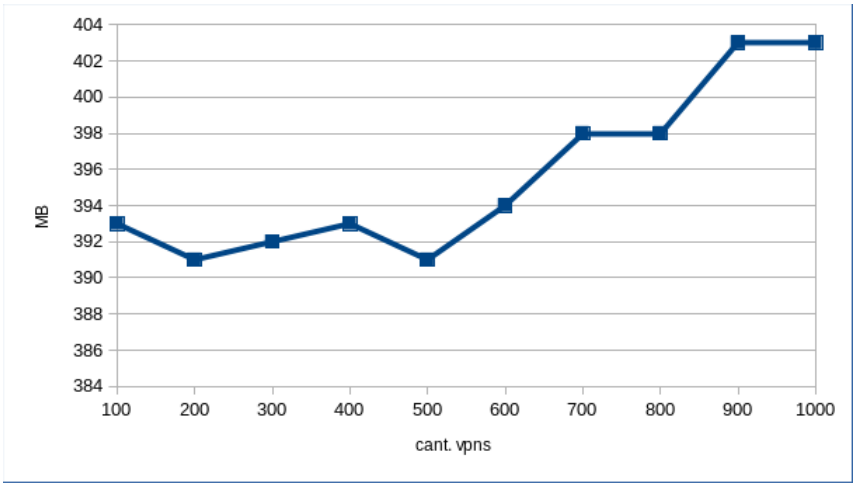

Fig. 8: Memoria utilizada en el plano de control en los routers PE. Gráfico generado con Libreoffice Calc

5.3. VPNs capa 3

Se corrió el siguiente script para realizar los tests de configuración de VPNs de capa 3:

```
#!/bin/bash

for i in $(seq 1 10)
do
        ./complete_cycle.sh 10 $(( $i*100 )) 5 3 0
done
```

donde los parámetros se establecen en:

- 10: cada test debe ejecutarse "10" veces
- $(($i*100)): se deben configurar "$(($i*100))" VPNs
- 5: cada VPN debe configurarse entre 1 y "5" routers
- 3: deben configurarse VPNs de capa "3"
- 0: se debe ejecutar cada test sin partir de ninguna medición anterior.

Luego de su finalización, para obtener los promedios de los tiempos de ejecución, sin considerar el primer ciclo de cada una de ellas se ejecutó la consulta:

```
select
        cant_vpns,
        round(avg(c_commit)::numeric,0) as c_commit,
        round(avg(c_total)::numeric,0) as c_total,
        round(avg(d_commit)::numeric,0) as d_commit,
        round(avg(d_total)::numeric,0) as d_total
from
        times t
where
        cycle>1 and
        nr_run=1 and
        layer=3
group by
        cant_vpns
order by
        cant_vpns asc;
```

Se obtuvieron los siguientes resultados:

cant_vpns	c_commit	c_total	d_commit	d_total
100	35	129	2	66
200	126	243	2	71
300	183	295	2	78
400	238	359	2	85
500	293	427	2	91
600	324	467	2	95
700	297	462	2	101
800	349	536	2	102
900	444	645	2	107
1000	511	727	2	113

Tab. 6: Resultado de los tests de capa 3

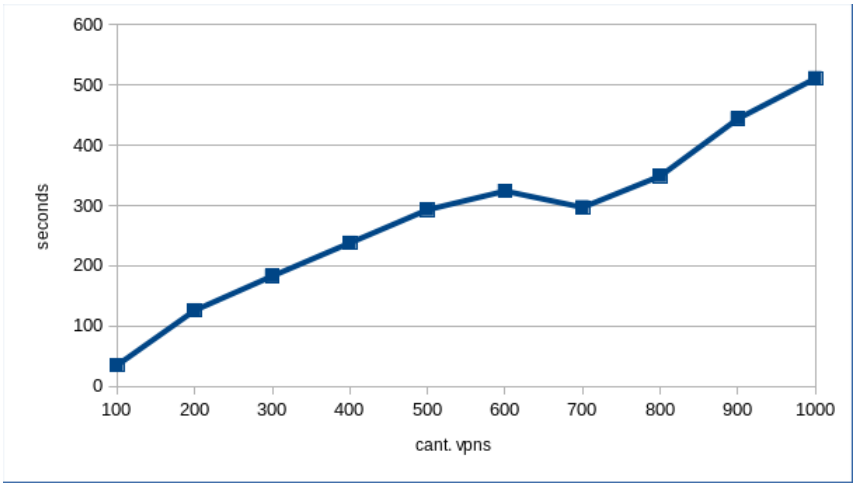

Fig. 9: Tiempo de creación de las VPNs de capa 3. Gráfico generado con Libreoffice Calc

Allí se aprecia claramente que el tiempo que demora en la creación de las VPNs crece casi linealmente a la cantidad de VPNs.

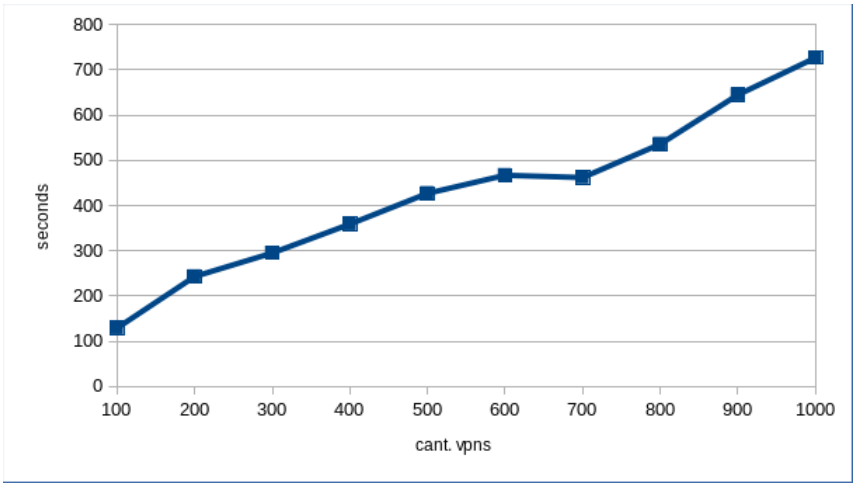

Fig. 10: Tiempo de creación de las VPNs de capa 3 incluyendo la configuración. Gráfico generado con Libreoffice Calc

Lo mismo sucede con el tiempo total, si bien la cantidad de segundos es mayor, el comportamiento es el mismo.

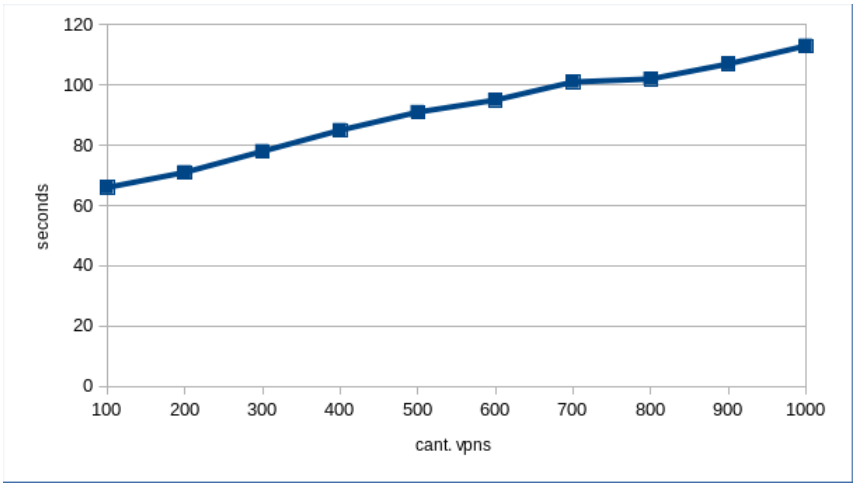

Fig. 11: Tiempo de borrado de las VPNs de capa 3. Gráfico generado con Libreoffice Calc

Aquí se observa que el tiempo de eliminación de las VPNs crece casi linealmente con la cantidad de VPNs.

Para obtener el consumo de memoria en los routers PE luego de la medición de los tiempos anteriormente descriptos se ejecutó la consulta:

```
select
        cant_vpns,
        round(total_memory) as total,
        round(control_plane_memory) as control_plane
from
        memory_usage
where
        layer=3
order by
        cant_vpns;
```

Se obtuvieron los siguientes resultados:

cant_vpns	total	control _plane
100	532	345
200	532	345
300	553	357
400	553	357
500	571	367
600	573	368
700	573	369
800	573	380
900	573	380
1000	594	391

Tab. 8: Resultado de las mediciones de memoria de capa 3

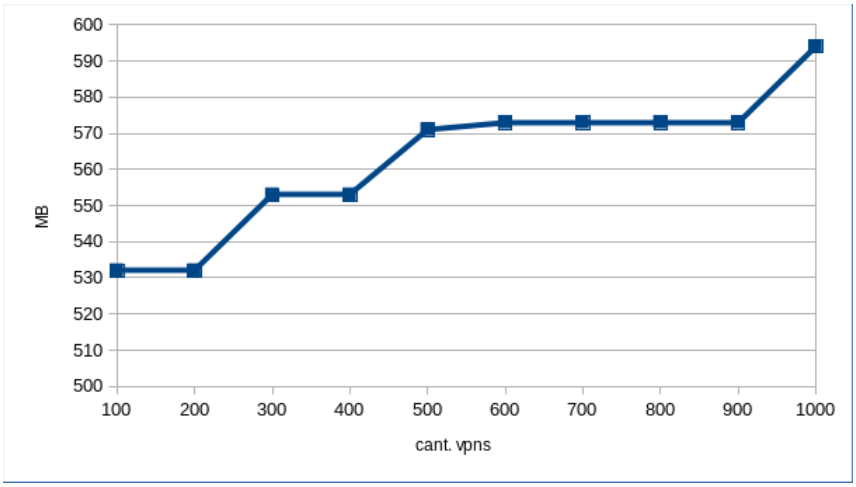

Fig. 12: Memoria total utilizada en los routers PE. Gráfico generado con Libreoffice Calc

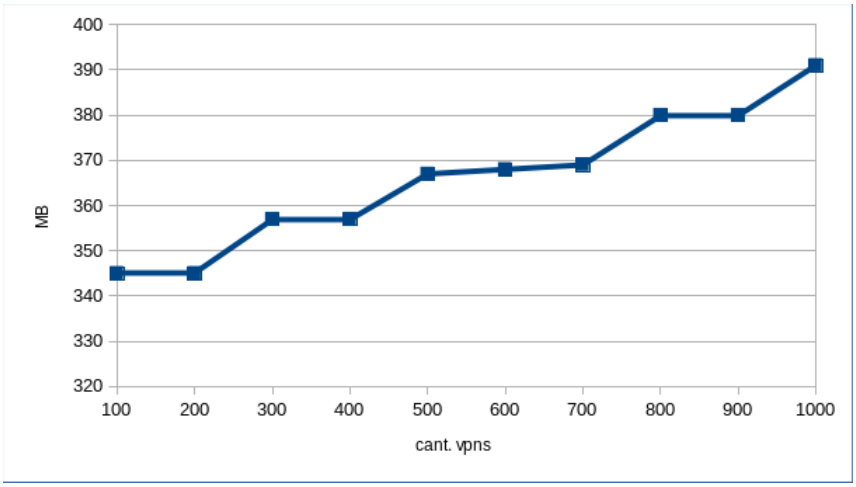

Fig. 13: Memoria utilizada en el plano de control de los routers PE. Gráfico generado con Libreoffice Calc

Según se puede apreciar en los resultados anteriores el tiempo se incrementa al ir agregando más y más VPNs de forma directamente proporcional a la cantidad de VPNs creadas, con un crecimiento muy cercano a lo lineal, tanto en la creación de las VPNs de capa 2 como las de capa 3.

Por otro lado vemos que la memoria consumida por los routers PE se incrementa al configurar un numero creciente de VPNs tanto en el caso de capa 2 como en el de capa 3. Y en ambos casos además al configurar 1000 VPNs el consumo de memoria está por debajo del 30 % de la memoria total.

Los resultados obtenidos en los tests confirman nuestras suposiciones respecto al comportamiento del tiempo y consumo de memoria al incrementar la cantidad de VPNs configuradas.

6. Conclusión y trabajo a futuro

Construimos un laboratorio emulado con 62 routers industriales que tienen soporte para MPLS, creamos herramientas para la configuración y eliminación automática de VPNs de capa 2 y capa 3, y finalmente realizamos mediciones del tiempo y memoria consumida por los routers al configurar y eliminar automáticamente 100, 200, 300, ... y 1000 VPNs de capa 2 y capa 3 sucesivamente.

Al analizar los resultados de las pruebas se puede observar que el comportamiento del tiempo de configuración y del consumo de memoria al utilizar la tecnología MPLS en la creación las VPNs es aceptable en relación al número de VPNs creadas tanto en las pruebas de capa 2 como en las de capa 3.

Estos resultados nos demuestran la gran capacidad de escala que posee esta tecnología ya que soporta la configuración de cientos de VPNs con un creci-

miento lineal del tiempo de configuración y del consumo de memoria de los dispositivos involucrado en relación a la cantidad de VPNs.

Como trabajo a futuro se buscará homogeneizar el ambiente de pruebas utilizado en este laboratorio de forma de tener una base de comparación correcta que pueda ser utilizada para comparar los resultados de [6] con los de este trabajo.

También como trabajo a futuro se podrían portar las herramientas creadas a un ambiente real para así obtener mediciones reales, o se podría utilizar un router Juniper real y configurar allí un número predeterminado de VPNs, repitiendo el procedimiento en uno de los routers emulados. De esa manera se obtendría una base de comparación entre ambos ambientes y se podrían sacar conclusiones sobre el tiempo y los recursos que podría consumir la creación de las VPNs del laboratorio en un entorno real no emulado.

A. Guía para la reproducción de los experimentos

Se entregan junto a este informe una serie de scripts para la automatización de las tareas de

- creación de la topología
- configuración de la red
- creación y borrado de las VPN
- medición de los tiempos
- chequeo de las interfaces de los routers
- corrección de las interfaces de los routers
- reinicio de los routers
- etc.

Dentro de la carpeta "scripts" están los scripts referentes a los experimentos del trabajo actual y sus resultados (carpetas con formato de fecha-hora, por ejemplo "201701311236"). También hay otras dos sub-carpetas, una con la herramienta para la creación de la topología ("topology_maker") y otra con los scripts para poner a punto el laboratorio para las pruebas ("routers_managment"). En el siguiente cuadro se detalla la estructura de carpetas:

```
.
├── commit_complete.sh
├── commit_router.sh
├── commit.sh
├── complete_cycle_fixer.sh
├── complete_cycle.sh
├── delete_vpnl2_ce_router.sh
├── delete_vpnl2_complete.sh
├── delete_vpnl2.sh
├── delete_vpnl3_complete.sh
├── delete_vpnl3.sh
├── enter_configuration_mode_router.sh
├── enter_configuration_mode.sh
├── get_memory_usage.sh
├── get_memory_usage_of_run.sh
├── global_definitions.sh
├── leave_configuration_mode_router.sh
├── leave_configuration_mode.sh
├── load_configuration.sh
├── random_routers_selection.sh
├── rollback_router.sh
├── routers_managment
```

```
    │   ├── check_interfaces.sh
    │   ├── configure_network.sh
    │   ├── configure_network_ce_ip_address_interface.sh
    │   ├── configure_network_pe_ip_address_interface.sh
    │   ├── configure_network_pep_ip_address_interface.sh
    │   ├── configure_network_p_ip_address_interface.sh
    │   ├── enter_cli_routers.sh
    │   ├── get_ports_from_gns3_file.sh
    │   ├── global_definitions.sh
    │   ├── halt_routers.sh
    │   ├── kill_all_expect_telnet.sh
    │   ├── reboot_router.sh
    │   ├── restart_all_routers.sh
    │   ├── restart_ce_routers.sh
    │   ├── restart_pe_routers.sh
    │   ├── restart_p_routers.sh
    │   ├── restart_router.sh
    │   ├── restart_routers_not_responding.sh
    │   ├── resume_all_routers.sh
    │   ├── resume_ce_routers.sh
    │   ├── resume_pe_routers.sh
    │   ├── resume_p_routers.sh
    │   ├── resume_router.sh
    │   ├── rise_up_interfaces_ce_routers_complete.sh
    │   ├── rise_up_interfaces_ce_routers.sh
    │   ├── router_ports.sh
    │   ├── suspend_all_routers.sh
    │   ├── suspend_ce_routers.sh
    │   ├── suspend_pe_routers.sh
    │   ├── suspend_p_routers.sh
    │   ├── suspend_router.sh
    │   └── wait_for_routers.sh
    ├── save_configuration.sh
    ├── set_vpnl2_ce.sh
    ├── set_vpnl2_pe_mpls_neighbor.sh
    ├── set_vpnl2_pe.sh
    ├── set_vpnl2.sh
    ├── set_vpnl3_ce.sh
    ├── set_vpnl3_pe.sh
    ├── set_vpnl3.sh
    ├── tests_1.sh
    ├── timer_deletel2.sh
    ├── timer_deletel3.sh
    ├── timer_setl2.sh
    ├── timer_setl3.sh
    ├── topology_description.txt
    ├── topology_description_p.txt
```

```
├── topology_description_pe.txt
├── topology_maker
│   ├── definition.h
│   ├── files_writer.c
│   ├── links_creation.c
│   ├── Makefile
│   ├── routers_creation.c
│   └── topology_maker.c
└── wait_for_routers.sh
2 directories, 76 files
```

Para reproducir los experimentos nos vamos a basar en el proyecto JunOSv62.gns3 creado con la herramienta "topology_maker" para versiones iguales o posteriores a la 1.2.3 de GNS3.

A.1. Iniciar el laboratorio

Para poder ejecutar las pruebas primero hay que obtener la topología del laboratorio y crear los archivos de configuración "topology_description.txt" y "router_ports.sh". Si se opta por utilizar las herramientas brindadas en el laboratorio se debe:

1. Ingresar en la carpeta "topology_maker", compilar la herramienta con "make" y luego ejecutarla con el comando "./topology_maker JunOSv62 62 36" para crear una topología con 62 routers, de los cuales 36 son CE, 12 PE y 14 P, interconectados de la manera que se explica en la sección "Requisitos de la topología". Luego hay que mover la carpeta "JunOSv62" generada a la carpeta "$HOME/GNS3/projects". Recordar que hay que editar el archivo "JunOSv62/JunOSv62.gns3" para reemplazar en cada caso necesario la ruta del ejecutable del emulador Qemu y la ruta de las imágenes .img de los routers.

2. Ingresar en la carpeta "routers_managment" y ejecutar la herramienta "get_ports_from_gns3_file.sh" cuyo único parámetro es el nombre del proyecto recién creado, en nuestro ejemplo "JunOSv62". Ésta herramienta genera el archivo "router_ports.sh".

3. Crear los archivos "topology_description.txt", "topology_description_pe.txt" y "topology_description_p.txt" manualmente según se explica en la sección "Requisitos de la topología" o copiar el que se da como ejemplo en esa sección que corresponde al proyecto "JunOSv62" utilizado en las pruebas.

inicial los routers del laboratorio. Para ello hay que:

1. Iniciar el servidor de GNS3:
   Hay que ejecutar el comando:

```
nohup gns3server --local > gns3server.out &
```

De esa manera el servidor de GNS3 queda corriendo en segundo plano (como servicio), escuchando en el puerto 8000 y escribiendo el log en gns3server.out.

2. Iniciar el servidor PostgreSQL:
   Hay que ejecutar el comando:

```
pg_ctl -D $HOME/pgdatadir -l $HOME/pgdatadir/logfile
    start
```

3. Iniciar los routers:
   Hay que iniciar el GNS3 de manera que luego de iniciar los routers el programa se pueda cerrar logrando que los routers sigan funcionando. Para lograr eso hay que abrir una nueva consola e iniciar el programa. Una vez iniciado hay que asegurarnos de que no intente iniciar él el servidor sino que utilice el que ya hemos iniciado manualmente, así que vamos al menú "Edit->Server preferences", a la sección "Server", deshabilitamos "Enable local server" y en la pestaña "Remote servers" agregamos al servidor que está corriendo en el host 127.0.0.1, en el puerto 8000. Luego abrimos el proyecto JunOSv62.gns3 e iniciamos los routers (menú "Control->Start all devices"). Una vez que estén corriendo los routers cerramos la consola de forma abrupta para que el proceso gns3 del programa finalice sin llegar a enviar la orden al servidor gns3server de detener la emulación de los routers.

4. Podemos verificar que estén funcionando iniciando un "telnet 127.0.0.1 2000" que es el router CE1 y ver que esté en proceso de inicio.

A.2. Configuración de las interfaces de los routers

Para poder configurar las interfaces de red de los routers (así como para todos los demás pasos del laboratorio), se debe haber ingresado al CLI de cada router. Para ingresar automáticamente en todos los routers hay que correr el script "enter_cli_routers.sh".

Luego debemos correr el script "configure_network.sh" que asigna las IP a las interfaces e inicia los protocolos necesarios en el resto del laboratorio.

A.3. Inicio de los tests

Para ejecutar los test vamos a utilizar el script "complete_cycle.sh" y lo ejecutaremos por ejemplo como "complete_cycle.sh 10 100 5 3 0" para iniciar un proceso de 10 corridas de creación y borrado de 100 VPNs de capa 3 con un máximo de 5 routers CE por VPN.

Luego de finalizada la ejecución se guardarán los resultados en la tabla "times"de la base de datos "tesis" del servidor PostgreSQL iniciado localmente, y se pueden obtener ejecutando por ejemplo la consulta:

```
select * from times where id=(select max(id) from times);
```

Para obtener los datos de la memoria consumida por los routers PE vamos a ejecutar el script "get_memory_usage.sh".

Luego de finalizada la ejecución se guardarán los resultados en la tabla "memory_usage" de la base de datos "tesis" del servidor PostgreSQL iniciado localmente, y se pueden obtener ejecutando por ejemplo la consulta:

```
select cant_vpns, round(total_memory) as total,
round(control_plane_memory) as control_plane,
round(data_plane_memory) as data_plane
from memory_usage where layer=3 order by cant_vpns;
```

B. Archivo de configuración "topology_description.txt" utilizado en el laboratorio

```
ce_id,pe_id,ce_if,pe_if
1,1,0,0
2,2,0,0
3,3,0,0
4,4,0,0
5,5,0,0
6,6,0,0
7,7,0,0
8,8,0,0
9,9,0,0
10,10,0,0
11,11,0,0
12,12,0,0
13,1,3,3
14,2,3,3
15,3,3,3
16,4,3,3
17,5,3,3
18,6,3,3
19,7,3,3
20,8,3,3
21,9,3,3
22,10,3,3
23,11,3,3
24,12,3,3
25,1,4,4
26,2,4,4
27,3,4,4
28,4,4,4
29,5,4,4
30,6,4,4
31,7,4,4
32,8,4,4
33,9,4,4
34,10,4,4
35,11,4,4
36,12,4,4
```

C. Archivo de configuración "topology_description_pe.txt" utilizado en el laboratorio

```
pe_id,p_id,pe_if,p_if
1,1,1,0
2,2,1,0
3,3,1,0
4,4,1,0
5,5,1,0
6,6,1,0
7,7,1,0
8,8,1,0
9,9,1,0
10,10,1,0
11,11,1,0
12,12,1,0
1,2,2,1
2,3,2,1
3,4,2,1
4,5,2,1
5,6,2,1
6,7,2,1
7,8,2,1
8,9,2,1
9,10,2,1
10,11,2,1
11,12,2,1
12,13,2,1
```

D. Archivo de configuración "topology_description_p.txt" utilizado en el laboratorio

```
p_id,p2_id,p_if,p2_if
1,2,2,3
2,3,2,3
3,4,2,3
4,5,2,3
5,6,2,3
6,7,2,3
7,8,2,3
8,9,2,3
9,10,2,3
10,11,2,3
11,12,2,3
12,13,2,3
13,14,2,3
14,1,2,3
1,4,4,5
2,5,4,5
3,6,4,5
4,7,4,5
5,8,4,5
6,9,4,5
7,10,4,5
8,11,4,5
9,12,4,5
10,13,4,5
11,14,4,5
12,1,4,5
13,2,4,5
14,3,4,5
```

E. Archivo de configuración "router_ports.sh" utilizado en el laboratorio

```
#!/bin/bash

CE1_IP=127.0.0.1
CE1_PORT=2000
PE1_IP=127.0.0.1
PE1_PORT=2001
P1_IP=127.0.0.1
P1_PORT=2002
CE2_IP=127.0.0.1
CE2_PORT=2003
PE2_IP=127.0.0.1
PE2_PORT=2004
P2_IP=127.0.0.1
P2_PORT=2005
CE3_IP=127.0.0.1
CE3_PORT=2006
PE3_IP=127.0.0.1
PE3_PORT=2007
P3_IP=127.0.0.1
P3_PORT=2008
CE4_IP=127.0.0.1
CE4_PORT=2009
PE4_IP=127.0.0.1
PE4_PORT=2010
P4_IP=127.0.0.1
P4_PORT=2011
CE5_IP=127.0.0.1
CE5_PORT=2012
PE5_IP=127.0.0.1
PE5_PORT=2013
P5_IP=127.0.0.1
P5_PORT=2014
CE6_IP=127.0.0.1
CE6_PORT=2015
PE6_IP=127.0.0.1
PE6_PORT=2016
P6_IP=127.0.0.1
P6_PORT=2017
CE7_IP=127.0.0.1
CE7_PORT=2018
PE7_IP=127.0.0.1
PE7_PORT=2019
P7_IP=127.0.0.1
P7_PORT=2020
CE8_IP=127.0.0.1
CE8_PORT=2021
PE8_IP=127.0.0.1
PE8_PORT=2022
P8_IP=127.0.0.1
P8_PORT=2023
```

```
CE9_IP=127.0.0.1
CE9_PORT=2024
PE9_IP=127.0.0.1
PE9_PORT=2025
P9_IP=127.0.0.1
P9_PORT=2026
CE10_IP=127.0.0.1
CE10_PORT=2027
PE10_IP=127.0.0.1
PE10_PORT=2028
P10_IP=127.0.0.1
P10_PORT=2029
CE11_IP=127.0.0.1
CE11_PORT=2030
PE11_IP=127.0.0.1
PE11_PORT=2031
P11_IP=127.0.0.1
P11_PORT=2032
CE12_IP=127.0.0.1
CE12_PORT=2033
PE12_IP=127.0.0.1
PE12_PORT=2034
P12_IP=127.0.0.1
P12_PORT=2035
CE13_IP=127.0.0.1
CE13_PORT=2036
CE14_IP=127.0.0.1
CE14_PORT=2037
CE15_IP=127.0.0.1
CE15_PORT=2038
CE16_IP=127.0.0.1
CE16_PORT=2039
CE17_IP=127.0.0.1
CE17_PORT=2040
CE18_IP=127.0.0.1
CE18_PORT=2041
CE19_IP=127.0.0.1
CE19_PORT=2042
CE20_IP=127.0.0.1
CE20_PORT=2043
CE21_IP=127.0.0.1
CE21_PORT=2044
CE22_IP=127.0.0.1
CE22_PORT=2045
CE23_IP=127.0.0.1
CE23_PORT=2046
CE24_IP=127.0.0.1
CE24_PORT=2047
CE25_IP=127.0.0.1
CE25_PORT=2048
CE26_IP=127.0.0.1
CE26_PORT=2049
CE27_IP=127.0.0.1
CE27_PORT=2050
CE28_IP=127.0.0.1
CE28_PORT=2051
```

```
CE29_IP=127.0.0.1
CE29_PORT=2052
CE30_IP=127.0.0.1
CE30_PORT=2053
CE31_IP=127.0.0.1
CE31_PORT=2054
CE32_IP=127.0.0.1
CE32_PORT=2055
CE33_IP=127.0.0.1
CE33_PORT=2056
CE34_IP=127.0.0.1
CE34_PORT=2057
CE35_IP=127.0.0.1
CE35_PORT=2058
CE36_IP=127.0.0.1
CE36_PORT=2059
P13_IP=127.0.0.1
P13_PORT=2060
P14_IP=127.0.0.1
P14_PORT=2061

CANT_ROUTERS_CE=36
CANT_ROUTERS_PE=12
CANT_ROUTERS_P=14
```

F. Hardware de los routers Juniper

Hardware de Juniper Olive

```
Del proceso de inicio de JunOS obtenemos:
JUNOS 10.1R1.8 #0: 2010-02-12 17:15:05 UTC
CPU: QEMU Virtual CPU version 2.3.0 (3058.17-MHz 686-class CPU)
Origin = "AuthenticAMD"  Id = 0x663  Stepping = 3
real memory  = 1073610752 (1023 MB)
avail memory = 1038823424 (990 MB)
```

Hardware de Juniper vSRX

```
root> show chassis hardware detail
Hardware inventory:
Item             Version  Part number  Serial number
    Description
Chassis                                                  FIREFLY-
    PERIMETER
Midplane
System IO
Routing Engine                                           FIREFLY-
    PERIMETER RE
  ad0     2047 MB  QEMU HARDDISK         QM00001         Hard Disk
FPC 0                                                    Virtual
    FPC
  PIC 0                                                  Virtual GE
Power Supply 0
```

```
root> show chassis routing-engine
Routing Engine status:
    Total memory              2048 MB Max   512 MB used ( 25
        percent)
      Control plane memory    1150 MB Max   334 MB used ( 29
          percent)
      Data plane memory        898 MB Max   189 MB used ( 21
          percent)
    CPU utilization:
      User                       2 percent
      Background                 0 percent
      Kernel                     1 percent
      Interrupt                  0 percent
      Idle                      97 percent
    Model                        FIREFLY-PERIMETER RE
    Start time                   2017-02-10 17:47:34 UTC
    Uptime                       58 minutes, 36 seconds
    Last reboot reason           Router rebooted after a normal
        shutdown.
    Load averages:               1 minute   5 minute  15 minute
                                     0.00       0.04       0.16
```

Tabla de figuras

Referencias

[1] The OpenStack project. OpenStack. https://www.openstack.org/.

[2] Linux Fundations Collaborative Projects. OpenDaylight. https://www.opendaylight.org/.

[3] GNS3 Technologies Inc. GNS3. https://www.gns3.com/.

[4] Qemu Project. Qemu. http://www.qemu-project.org/.

[5] Francesco Palmieri. VPN scalability over high performance backbones evaluating MPLS VPN against traditional approaches.

[6] Santiago Vidal, Jorge Rodrigo Amaro, Emiliano Viotti, Martin Giachino, and Eduardo Grampin. RAUflow: building virtual private networks with MPLS and OpenFlow. In Proceedings, LANCOMM 16, workshop on Fostering Latin-American Research in Data Communication Networks, Pages 25-27. Florianopolis, Brazil - August 22 - 26, 2016, ACM New York, NY, USA 2016. ISBN: 978-1-4503-4426-5 http://dl.acm.org/citation.cfm?id=2940133.

[7] Mark Lewis. *Comparing, Designing, And Deploying VPNs.* Cisco Press.

[8] Network Working Group. RFC 3031 - multiprotocol label switching architecture. https://tools.ietf.org/rfc/rfc3031.txt.

[9] Luc De Ghein. *MPLS Fundamentals.* Cisco Press.

[10] Network Working Group. RFC 2547 - BGP/MPLS VPNs. https://tools.ietf.org/html/rfc2547.

[11] Tech Library, Juniper Networks. Introduction to VPLS. https://www.juniper.net/documentation/en_US/junos12.3/topics/concept/vpn-vpls-introduction.html.

[12] Tech Library, Juniper Networks. VPN routing and forwarding tables. http://www.juniper.net/documentation/en_US/junos12.3/topics/concept/vpn-routing-tables-vpn-forwarding-tables.html.

[13] Cisco Systems, Inc. Cisco 7200 series routers. http://www.cisco.com/c/en/us/support/routers/7200-series-routers/tsd-products-support-series-home.html.

[14] Cisco Systems, Inc. Cisco. http://www.cisco.com.

[15] Juniper Networks. Juniper. http://www.juniper.net/us/en/.

[16] Juniper Networks. Juniper vSRX virtual firewall. http://www.juniper.net/us/en/products-services/security/srx-series/vsrx/.

[17] Junper Networks. Routing instances overview. https://www.juniper.net/documentation/en_US/junos/topics/concept/routing-instances-overview.html.

[18] Bruno Quoitin, Virginie Van den Schrieck, Pierre Francois, and Olivier Bonaventure. IGen: Generation of router-level internet topologies through network design heuristics. https://inl.info.ucl.ac.be/system/files/itc21-igen-bqu.pdf.

[19] Network Information Library. How to run JunOS inside of GNS3 - step by step. http://nil.uniza.sk/network-simulation-and-modelling/gns3/how-run-junos-inside-gns3-step-step.