PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO

Escuela de Ingeniería Eléctrica
PONTIFICIA UNIVERSIDAD CATOLICA DE VALPARAISO

Víctor Manuel Arriagada Peña

Estudio de software Cisco VIRL y sus potenciales aplicaciones en redes de datos

Informe Proyecto de Título de Ingeniero Civil Electrónico

Escuela de Ingeniería Eléctrica

PONTIFICIA
UNIVERSIDAD
CATÓLICA DE
VALPARAÍSO

Estudio de software VIRL (Virtual Internet Routing Lab) de Cisco y sus potenciales aplicaciones en redes de datos

Víctor Manuel Arriagada Peña

Informe Final para optar al título de Ingeniero Civil Electrónico,
aprobada por la comisión de la
Escuela de Ingeniería Eléctrica de la
Pontificia Universidad Católica de Valparaíso
conformada por

Sr. Francisco Alonso Villalobos
Profesor Guía

Sr. Felipe Leighton González
Segundo Revisor

Sr. Sebastián Fingerhuth Massmann
Secretario Académico

Valparaíso, 21 de junio de 2017

Dedicado a mi familia, Clotilde Maricela Fierro Verdejo, Nilsa Peña Fierro,

Fernanda Lorca San Martin, Javiera Arriagada Peña,

Rigoberto Peña Verdejo, Victor Arriagada Madrid,

Sebastián Arriagada Peña, Tíos y primos.

Agradecimientos

Formar un Profesional es tarea difícil, es un trabajo en equipo donde muchas personas colocan su energía en un solo individuo, o la distribuyen hacia un grupo en el caso de los distinguidos Profesores.

Agradezco a mi familia, quienes siempre me apoyaron en mis decisiones y confiaron en mí. Agradezco a mi polola, quien ha sido una parte muy importante en este proceso. Agradezco a mis compañeros y amigos de la universidad, amigos de la vida. Agradezco a los Profesores por compartir y mostrar sus conocimientos, sin ellos no sería posible esta misión.

Por último, agradezco a Valparaíso y su gente por acogerme durante este proceso, creo que el medioambiente influye, es parte de la integración.

Valparaíso, 21 de junio de 2017 V. M.

Resumen

Conocer las capacidades de una herramienta que permita simular topologías de red de manera fiel y confiable, manteniendo costos bajos de implementación, con características que permitan poseer un ambiente de laboratorio, no hace otra cosa que aportar en el área de las redes de datos. Cisco es la empresa líder en el área, quien ha desarrollado un software con una amplia variedad de características.

La hipótesis inicial, es que el software Cisco VIRL, posee las mejores características y desempeño para realizar simulaciones de redes lo más semejante a trabajar con dispositivos reales. Hipótesis que será corroborada con el estado del arte.

Para tener certeza y conocer todas las características del software, además de aprender a manipularlo, se genera una guía que comienza desde los requerimientos de software y hardware necesario para su implementación, pasando por la instalación y puesta en marcha, familiarización del entorno de trabajo con el usuario, simulaciones de topologías de red y sus configuraciones, hasta una manipulación avanzada.

Esto permitirá conocer las competencias del software en paralelo con su manipulación, dejando claridad sobre sus capacidades y sus potenciales aplicaciones en redes de datos. Sin embargo, un análisis económico, confirmará la conveniencia de su implementación.

Palabras claves: Cisco VIRL, características de Cisco VIRL, simulación redes de datos, experiencias de laboratorio con Cisco VIRL, Imágenes Cisco virtualizadas.

Abstract

Knowing the capabilities of a tool that allows to simulate network topologies in a faithful and reliable way, maintaining low implementation costs, with characteristics that allow to have a laboratory environment, only comes to contribute in the area of data networks. Cisco is the leading company in the area, who developed software with a wide variety of features.

The initial hypothesis is that Cisco VIRL software has the best features and performance to perform network simulations, much like working with real devices. This hypothesis will be corroborated with the state of the art.

In order to be sure and to know all the characteristics of the software, in addition to learning to manipulate it, a guide is generated starting from the softwares and hardware requirements for its implementation, through installation and commissioning, familiarization of the work environment with the user, simulations of network topologies and their configurations, to an advanced manipulation.

This will allow to know the competences of the software in parallel with its manipulation, leaving clarity on its capacities and its potential applications in data networks. However, an economic analysis will confirm the desirability of its implementation.

Key words: Cisco VIRL, features of Cisco VIRL, simulation data networks, lab experiences whit Cisco VIRL, virtualized Cisco images.

Índice general

Índice de figuras

Índice de tablas

Introducción

La creciente demanda de proveer nuevos servicios de red, eficaz y eficientemente, conduce a la necesidad de diseñar, realizar pruebas e implementar redes rápidamente. Realizar pruebas a escala con equipamiento dedicado, resulta ser todo un desafío. Puesto que se hace necesario la implementación de un laboratorio con equipamiento de alto costo, necesidades de refrigeración y por tanto un alto consumo de energía eléctrica, además de horas hombre [1].

Llevar a cabo cambios de red es fundamentalmente difícil. Esto se debe principalmente al hecho de que las redes modernas son compartidas inherentemente, y cualquier cambio en la red tiene el potencial de impactar negativamente a los usuarios y los servicios existentes [1].

El sector de educación enseña los principios y conceptos de redes a sus estudiantes. El costo de los equipos de redes de calidad comercial conduce a mantener un gran número de estudiantes por equipo, lo que limita el tiempo en que cada estudiante se relacione con ellos. Cubrir estos costos se torna complejo o simplemente no se puede incurrir en ellos, sin embargo, las habilidades y el conocimiento son esenciales para el crecimiento de la economía local de Tecnología de la Información.

Encontrar una herramienta que facilite realizar pruebas, crear y diseñar redes, ya sea con fines empresariales o educativos, facilitaría el acceso y disminuiría los montos iniciales de inversión. Debido a esto, se realiza el estudio del software Cisco VIRL, captando las capacidades que posee éste en relación a las necesidades planteadas.

El problema se aborda realizando un estudio sobre el estado del arte respecto de software que funcionan en entornos virtuales, o que poseen capacidades similares en cuanto a una serie de características como la de programar comandos y/o automatizar simulaciones, apuntando hacia un entorno amigable para el usuario. Lo anterior se realiza teniendo en mente que la empresa Cisco Systems es líder mundial en el área [2] y por tanto podría surgir la pregunta ¿Si la empresa Cisco Systems es líder mundial en el área de redes de datos, por qué el software VIRL podría tener competencia? A priori, esta pregunta poseería una respuesta positiva o negativa. Pero el objetivo principal no es saber qué software es mejor según ciertos parámetros o criterios, sino se comienza a partir de la hipótesis que el software Cisco VIRL posee las mejores características para cubrir las necesidades planteadas anteriormente. Sin embargo, el estudio sobre el estado del arte confirmará esta hipótesis.

Luego de concluir con el estudio sobre el estado del arte, se comprenderá que VIRL es superior en varios sentidos, compitiendo solamente con el software GNS3. El foco se centra sólo en las capacidades del software a partir de ese momento. Comenzando por reconocer el entorno de trabajo como máquina virtual, mediante la selección de las herramientas más apropiadas para la virtualización, recurso imprescindible que será necesario para el correcto funcionamiento del software. Debido a que VIRL provee una gran cantidad de servicios para realizar simulaciones, es que se hace necesario el uso de determinada capacidad de hardware arquitectura x86 para su óptimo funcionamiento.

Una vez esquivados los primero obstáculos, el problema que surge a continuación, corresponde a cómo utilizar el software de la mejor manera posible, y luego formular un documento que facilite su entendimiento, pudiendo entregar habilidades como conocer los requerimientos de la puesta en marcha, además de fundamentos para una manipulación completa. Entonces, se plantea la utilización de una guía, la que comienza con la instalación, continuando con la manipulación, herramientas de simulación, y todo lo que respecta al funcionamiento del software, sin dejar de mostrar y comprender la manera en que opera sobre el PC. Así, se adquirirán los conocimientos básicos para su correcta manipulación. La dificultad de esta guía comienza con un nivel muy bajo, aumentando paulatinamente, con el objetivo de que luego de leer y desarrollar los ejercicios planteados en este informe, el usuario quede completamente capacitado para utilizar el software, teniendo pleno conocimiento del máximo de sus capacidades y obteniendo la autonomía para su manipulación.

VIRL necesita tener acceso a entornos virtuales de red [1] [3] [4], y es esta razón por la que instalar el software no es algo trivial, ya que se deben realizar configuraciones de redes virtuales, las que el software utilizará para poder ejercer el manejo de las simulaciones y también para poder conectarse con redes externas. Se describe un serie de instrucciones que guían la instalación del software hasta quedar habilitado y funcionando.

Un primer acercamiento al software será ilustrado en una serie de experiencias prácticas, guiadas paso a paso, partiendo por el entorno gráfico y el modo de interacción con el usuario, reconociendo ventanas, botones, menús, además del contenido y acción de cada cual. Esta primera parte introductoria, concluye con la puesta en marcha de una simulación simple.

Una segunda sección es incluida con experiencias avanzadas donde se conocerán características y opciones propias de las simulaciones.

Finalmente se presenta una tercera sección abordando temas relacionados a las capacidades del software. Por ejemplo, la automatización de funciones en una simulación por medio de herramientas tales como Python. Además, se expone la manera en que el software realiza las simulaciones mediante la utilización de APIs (*Application Programming Interfaces*, Interfaz de Programación de Aplicaciones en Español). Se agrega una guía con fundamentos para configurar y conectar el software con redes reales. Por último, una ficha de laboratorio concluye con una actividad para conocer características avanzadas del software, como la inyección de latencia, pérdida de paquetes, entre otros.

Sin embargo, una vez conocidas las capacidades de Cisco VIRL, sus características y la manera de operarlo, se estaría respondiendo parcialmente a las necesidades planteadas. Se hace necesario realiza un estudio técnico y económico para la implementación del software. Así, se pondrá en evidencia la certeza sobre la conveniencia económica en poner en marcha el software, en contraste con un laboratorio con equipos de calidad comercial, o la utilización de ambos.

Es decir, luego de tener claridad sobre los pro y contras del software respecto de equipamiento real, como disponibilidad de características en las variaciones de las imágenes Cisco (IOS, IOS-XE, IOS-XR…), además de estudiar la posibilidad de simular sistemas de otros propietarios, surge la interrogante si implementar el software, en términos económicos, resulta más conveniente que utilizar equipos reales, ya sea para fines comerciales o educativos.

1 Antecedentes generales y solución propuesta

1.1 Descripción del problema

Las redes de comunicación de datos son complejas, trabajan en ambientes integrados, donde los grupos de operaciones poseen el desafío de mantener la estabilidad de los sistemas mientras se implementan nuevos servicios o se realizan cambios. Los ingenieros que diseñan e implementan redes, deben considerar los nuevos requerimientos de servicios y la evolución controlada de estos [1]. Shadownets [5], presenta una solución, donde se construyen redes en paralelo para facilitar la realización de pruebas para validar cambios. Esto es una solución, sin embargo, resulta muy costosa debido a su naturaleza.

El sector de educación enseña los principios y conceptos de redes a sus estudiantes. El costo de los equipos de redes de calidad comercial conduce a mantener un gran número de estudiantes por equipo, lo que limita el tiempo en que cada uno se relacione con ellos. En muchas partes del mundo, cubrir estos costos se torna complejo o simplemente no se puede incurrir en ellos, sin embargo, las habilidades y el conocimiento son esenciales para el crecimiento de la economía local de TI (tecnología de la información). Interactuar con la práctica para la creación de redes, mejora las habilidades y la retención del conocimiento [1].

En definitiva, el alto costo de los equipos que se requieren, ya sea para fines educativos o para probar y validar cambios, con el objetivo de reducir el riesgo de afectar a las redes en funcionamiento, han dado lugar a la limitada disponibilidad de acceso a entornos de prueba. Los ingenieros necesitan soluciones rápidas, soluciones como por ejemplo, que se disponga de equipos cuando se requieran y no cuando estén disponibles.

1.2 Estado del arte

En esta sección se dará a conocer softwares que permiten simulación, emulación y virtualización de redes de datos.

En la actualidad, las compañías necesitan ser cada vez más ágiles, adelantarse a su competencia con nuevos negocios, productos y/o servicios, tomando ventaja de las nuevas tecnologías como IoT (*Internet of Things*, Internet de las cosas en Español), SDN (*Software design Networking*, redes definidas por Software en Español), y *Cloud Computing* (Computación en la nube), especialmente cuando sus productos se comercializan a través de internet. Para ello las áreas de TI son fundamentales. Ellas proveen de la infraestructura donde se instalarán estos nuevos servicios, ya sea con nuevo equipamiento o aumentando la capacidad de la red existente [6].

Software que permitan la simulación, emulación y virtualización de redes de datos son recursos valiosos que entregan flexibilidad, tiempo y ahorro en recursos económicos.

1.2.1 Conceptos

- **Simulación.**

  Un simulador es un sistema de software que imita otro sistema complejo, con un nivel variable de realidad. El uso más común de simuladores de red es para fines de capacitación, sobre todo cuando es peligroso hacer la actividad sin entrenamiento o se necesitan probar nuevas configuraciones. Aunque la mayoría de los simuladores no suelen ser muy realistas, algunos incluyen cuestiones físicas y detalles de software [7].

- **Emulación y Virtualización.**
  Un emulador es un software que permite ejecutar programas en una plataforma, ya sea una arquitectura de hardware o un sistema operativo, es decir que utiliza una plataforma diferente para la cual fueron escritos originalmente. La diferencia con un simulador, es que este último sólo trata de reproducir el comportamiento del programa. Un emulador trata de modelar de forma precisa el dispositivo de manera que este funcione como si estuviera siendo usado en el aparato original. En cambio, en un entorno de virtualización, al software no le importan los recursos físicos o emulados donde se estén corriendo. La clave de la virtualización reside en el aislamiento de procesos en contenedores separados dentro del mismo hardware. Por ejemplo, un sistema operativo IOS puede ejecutarse virtualmente (sobre instancias de hardware Intel) o puede usarse emulando el hardware de la plataforma específica donde funciona en la vida real (emulando hardware Power PC) [7].

1.2.2 Softwares

A continuación, se presenta una lista de softwares que han sido seleccionados como herramientas que ayudarían con los problemas planteados. Se ilustra una breve descripción de cada uno de ellos para luego realizar una tabla comparativa con criterios y parámetros cualitativos bien definidos.

- **GNS3**

  GNS3 es un simulador gráfico de red que permite diseñar topologías de red complejas y poner en marcha simulaciones sobre ellas.

Para permitir completar simulaciones, GNS3 está estrechamente vinculado con:

- Dynamips, Dynagen [8].
- Qemu [9].
- VirtualBox [10].
- Wireshark [11].
- VPCS (Virtual PC Simulator) [12].

GNS3 es una excelente herramienta complementaria a los equipos físicos para los administradores de redes o las personas que quieren pasar sus certificaciones [7].

- **CORE [7]**

  Common Open Research Emulator (CORE) es una herramienta para emular redes en una o más máquinas virtuales. Basándose en Linux y Quagga [13].

  Las redes emuladas se pueden conectar a redes reales. CORE consiste de un GUI (interfaz gráfica de usuario en español) para dibujar topologías de máquinas virtuales livianas y módulos de Python para automatizar la emulación de las redes.

  CORE es un software de código abierto, liviano, además genera paquetes por lo que se puede interconectar con una red en servicio. No soporta características propietarios de algunos fabricantes, cada protocolo se gestiona desde una CLI (interfaz de línea de comando en español) diferente.

- **Cisco Packet Tracer [14]**

  Es un potente programa de simulación de red que permite a estudiantes experimentar y visualizar el comportamiento de la red, junto con el flujo de paquetes.

  Está disponible de forma gratuita a los instructores de *Networking Academy*, estudiantes, ex alumnos y administradores que estén registrados en Netspace.

  Este software permite visualizar cómo los paquetes se mueven en la red, y pueden ser analizados según cada capa del modelo OSI. Puede hacer simulaciones de servicios, tales como servidores WEB o DNS. Simula *access points* inalámbricos y teléfonos IP. Además, dado que no utiliza la virtualización, no requiere un PC potente. Hay una versión 'wine-izada' oficial para Linux. Permite varios usuarios colaborando en una misma simulación.

  Las simulaciones en Packet Tracer implementan equipos de gama baja (2811 / 2960) [7]. Está pensado para estudiar CCNA o aprender fundamentos básicos de redes. No implementa todo el sistema operativo de los equipos, por lo que no tiene todas las características habilitadas.

- **Mininet [15]**

  MiniNet es un emulador de red que crea una red de hosts virtuales, switches, controladores y enlaces. MiniNet ejecuta software de red estándar de Linux, y sus switches son compatibles con OpenFlow para enrutamiento personalizado altamente flexible y redes definidas por software (SDN).

  MiniNet apoya la investigación, el desarrollo, el aprendizaje, la creación de prototipos, pruebas, depuración, y cualesquiera otras tareas que podrían beneficiarse de tener una red experimental completa en un laptop o desktop.

- **Netkit [16]**

  Netkit es un entorno para la creación y la realización de experimentos de redes a bajo costo y con poco esfuerzo. Permite "crear" varios dispositivos de red (routers virtuales, switchs, hots, etc.) que pueden ser fácilmente conectados entre sí con el fin de formar una red en un único PC. Los equipos de rede son virtuales pero cuentan con muchas de las características de los reales, incluyendo la interfaz de configuración.

  Emular una red con Netkit es una cuestión de escribir un simple archivo que describe la topología de nivel de enlace de la red para ser emulado y algunos archivos de configuración que son idénticos a los utilizados por las herramientas de red del mundo real. Netkit se encarga de iniciar los dispositivos de red (emulados) y de la interconexión de los mismos según sea necesario. Alternativamente, las redes pueden ser descritas mediante el uso de un lenguaje basado en XML conocido como NETML. A partir de una descripción de la red en NETML, es posible obtener automáticamente archivos de configuración que se pueden utilizar con routers reales, o secuencias de comandos Netkit que pueden utilizarse para emular la red descrita.

1.2.3 Comparación de softwares

GNS3, CORE, Cisco Packet Tracer, MiniNet y Netkit, son los softwares seleccionados que brindan las facilidades para interactuar con simulaciones de redes de datos, los que poseen un ambiente de trabajo similar al que ofrece VIRL.
A continuación se presentan seis parámetros que ayudarán a posicionar estos softwares en una escala de características [17]. En cada caso, se define el parámetro y la manera cualitativa para su medición.

Tipo de licencia

Para efectos prácticos, dependiendo del tipo de licencia de la herramienta, ella puede tener habilitadas todas sus funciones o solo un grupo de ellas; esto también puede ir ligado con el valor que se cobre por la licencia. Con base en lo anterior, la forma de categorizar los tipos de licencias es:

- **Libre:** A partir de esta forma de cuantificación, se plantea que el software libre no es necesariamente una herramienta gratuita, sino que, en realidad, lo que la hace libre es la

posibilidad que ofrece a los usuarios para editarla, copiarla, ejecutarla, distribuirla, estudiarla y mejorarla.
- **Comercial:** Se refiere a las licencias que tienen restricciones para el usuario, teniendo en cuenta que su comercialización, costo, duración de uso, edición y libertad de permisos, son controlados y definidos por los propietarios de la herramienta. Este tipo de licencia también es conocida como licencia propietaria, en contrapartida al licenciamiento libre o software libre.

Plataformas que lo soportan

Este parámetro es descriptivo. Ilustra los diferentes sistemas operativos en los cuales la herramienta puede correr sin ningún problema. Los sistemas operativos que se tendrán en cuenta son: Windows, Linux, Mac.

Interfaz Gráfica

Con este parámetro se busca definir la cercanía que tiene la herramienta con el usuario y las facilidades que le presta. La medida de este parámetro tendrá en cuenta tres rangos:

- **Alto:** Requiere un nivel de programación mínimo, ya que la herramienta tiene la disposición de trabajar desde todas sus perspectivas con una interfaz gráfica.
- **Medio:** Implementa una interfaz gráfica que facilita su uso, pero lo hace de forma limitada; algunas de sus implementaciones deben definirse mediante programación.
- **Bajo:** La herramienta no cuenta con interfaz gráfica o ella no es muy amigable con el usuario, lo cual implica la programación de cada elemento dentro de una simulación para su ejecución final.

Graficación de resultados

Las herramientas de simulación se utilizan para recrear el funcionamiento de la red de la manera más real posible. Para ello es necesario realizar medidas de ciertas variables de red, con el fin de realizar un análisis posterior de los datos y comprender así el comportamiento de la red ante diferentes eventos o posibles configuraciones. Una manera de interpretar y analizar los datos de las variables medidas es graficándolos de diversas formas. Dependiendo qué tan potente o amigable sea la herramienta de simulación para realizar esta tarea se han definido los siguientes rangos:

- **Buena:** La herramienta posee extensiones o módulos propios para la generación graficas estadísticas las cuales pueden ser manipuladas desde el mismo simulador o ser exportadas a un procesador especializado.
- **Aceptable:** Aquellos simuladores que pueden generar datos estadísticos, pero que necesitan de una herramienta externa, para generarlos, procesarlos adecuadamente y presentar la información ordenada al usuario.
- **Limitada:** Aquellas herramientas que no cuentan con un módulo propio o extensión para la generación de gráficos; la información estadística puede estar representada en archivos

de texto que necesitan de herramientas diferentes a la de simulación para su organización y presentación.

Tecnologías y protocolos de niveles 2 y 3 que soporta

Parámetro de índole descriptivo en el cual se listan las tecnologías y protocolos de nivel 2 y 3 del Modelo OSI que soporta. Algunas herramientas no soportan todos los protocolos, lo que hace necesaria su implementación generando código o adaptando componentes preexistentes. De este modo se quiere clasificar cada software de acuerdo con la variedad de protocolos que permiten simular, con base en los siguientes criterios:

- **Alto:** Que permite la implementación de gran cantidad de tecnologías/protocolos de red, ya que posee módulos propios con la arquitectura necesaria para que sean soportados y desplegados de manera correcta, con el fin de acercarse a implementaciones reales.
- **Medio:** Simuladores que no permiten realizar implementaciones de un gran número de tecnologías/protocolos, puesto que no poseen los módulos necesarios o en su defecto es necesario modificar el código fuente de alguno de sus módulos para lograr simular el protocolo deseado.
- **Bajo:** Aquellas herramientas que no poseen los módulos de las tecnologías/protocolos desarrollados o en su defecto es necesario conseguir los módulos por separado.

Conectividad exterior

Este parámetro indica si es posible conectar el simulador con tráfico exterior. Es decir que el software tenga la posibilidad de generar paquetes y conectarse con una red en servicio.

Parametrización de softwares seleccionados

La tabla 1-1, presenta el resumen de cada Software respecto de los parámetros anteriores mencionados. Esta tabla no considera la discriminación por la utilización de tecnologías de virtualización.

GNS3 puede o no utilizar la virtualización como herramienta, a diferencia de Cisco VIRL quien lo utiliza en todo momento. Esto permite realizar simulaciones con imágenes de máquinas reales, asignando recursos de software por medio de algún hipervisor soportado, tanto para Cisco VIRL o GNS3.

Tabla 1-1. Parametrización de Softwares de simulación.

Software / Parámetro	GNS3	CORE	Packet Tracer	MiniNet	Netkit	VIRL
Tipo de licencia	Libre/ Comercial	Libre	Comercial	Libre	Libre	Comercial
Plataformas que lo soportan	Linux, Mac, Windows	Linux	Windows, Linux	Linux	Linux	Windows, Mac, Linux
Interfaz gráfica	Aceptable	Medio	Alto	Medio	Alto	Alto
Graficación de resultados	Limitada	Limitada	Limitada	Limitada	Limitada	Buena
Tecnología y protocolos de nivel 2 y 3 que soporta	Alto	Bajo	Medio	Bajo	Bajo	Alto
Conectividad exterior	Si	No	No	Si	Si	Si
Costo en USD	$0	$0	$0	$0	$0	$199.99 por año

La tabla 1-1, ilustra la suficiente información para considerar que GNS3, Cisco Packet Tracer y Cisco VIRL, son los tres softwares más apropiados para realizar simulaciones de redes de datos según los parámetros descritos anteriormente. Esto sin considerar que Cisco Packet Tracer no utiliza la Virtualización como principio de funcionamiento, ya que sólo trabaja con partes de los sistemas operativos Cisco incluidos, transformándose en una herramienta que no servirá para tener certeza sobre, por ejemplo, evaluar nuevas configuraciones en una red existente, ya que no hay una representación fiel de dicha red. A diferencia de GNS3 y VIRL, que al utilizar la virtualización como principio de funcionamiento, es posible operar máquinas virtuales con sistemas operativos Cisco como IOS, IOS-XE, etc., o la implementación de otras máquinas virtuales con sistemas operativos de otros propietarios, lo que permitirá ejecutar pruebas con una representación fiel, de una red de datos.

Virtualizar aporta ventajas y posibilidades únicas en la actualidad. Permite reducir costos en prácticamente todos los campos de actuación de la administración de sistemas; desde la instalación y configuración de equipos hasta los procesos de copias de seguridad, monitorización, gestión y administración de la infraestructura. Disminuye el número de servidores físicos necesarios y el porcentaje de desuso de los recursos que se disponen, aumentando su eficiencia energética. También brinda la posibilidad de centralizar y automatizar procesos cuya

administración normalmente consume mucho tiempo, pudiendo aprovisionar y migrar máquinas virtuales de una manera rápida y fiable, manteniendo buena la calidad del servicio y rápido tiempo de respuesta ante una caída del mismo [18]. Las técnicas de virtualización se pueden aplicar a otras capas de infraestructura de TI, incluyendo redes, almacenamiento, ordenador portátil o hardware de servidor, sistemas operativos y aplicaciones [19]. Esta es la principal razón por la que GNS3 y Cisco VIRL utilizan la virtualización como principio de funcionamiento [20], puesto que les permite lograr un alto desempeño.

1.3 Solución propuesta en base a lo presentado en el estado del arte

Dada la información entregada por la tabla 1-1 y el posterior análisis, GNS3 y Cisco VIRL corresponden a los softwares que presentan las mejores características de simulación, quienes podrían entregar herramientas para satisfacer las necesidades planteadas en 1.1. Para esto, se hace necesario realizar un estudio en profundidad para conocer las capacidades de los softwares. Sin embargo GNS3 no será estudiado, ya que según la tabla 1-1, Cisco VIRL es superior en interfaz gráfica y graficación de resultados. Otro punto a favor que posee Cisco VIRL respecto de GNS3, es que las imágenes de los sistemas operativos de Cisco Systems, como IOS, corresponden a los mismos creadores de VIRL, a diferencia de GNS3, donde se podría estar incurriendo en alguna falta legal debido a la utilización de propiedad intelectual [21]. Por tanto, el estudio se realizará sólo sobre Cisco VIRL.

1.4 Objetivos generales y específicos

- Objetivo general:
    - Estudiar las capacidades del software de diseño de redes Virtual Internet Routing Lab de Cisco, y analizar las sus potenciales competencias en redes de datos.
- Objetivos específicos:
    - Comprender y exponer el principio de funcionamiento del software, referente a su estructura y tecnología de virtualización.
    - Desarrollar una guía práctica por medio de experiencias de laboratorio, que permita conocer el software desde las configuraciones iniciales de instalación, hasta características avanzadas de simulación.
    - Estudio de características y herramientas de simulación que ayuden a realizar pruebas sobre topologías simuladas. Conectividad exterior.
    - Estudio de características disponibles en las imágenes IOSv, CSR1000v, IOS-XRv, NX-OSv y ASAv, respecto de las utilizadas en equipos reales.
    - Factibilidad de correr máquinas virtuales de otros propietarios en el software.
    - Realizar un análisis técnico y económico de la implementación del software.

1.5 Solución propuesta

Fijados los objetivos específicos, la solución propuesta para realizar el estudio del software, comienza por comprender el principio de funcionamiento, la manera en cómo opera sobre el PC. Específicamente la virtualización, ya que es el motor fundamental de las características que posee

VIRL [1], y será determinante a la hora de escoger el hipervisor más apropiado. Estudiar la estructura del software también es de interés, puesto que será clave para comprender conectividad exterior.

Luego de tener claridad sobre los requerimientos de software y hardware para el funcionamiento apropiado, es posible instalarlo. Para esto, se emplea una guía que ilustre con detalle los pasos a seguir, con apoyo en imágenes que facilitarán su comprensión y rápida implementación.

A continuación, surge la necesidad de aprender a utilizar y manipular el software. Por lo que se presentarán una serie de actividades por medio de fichas de laboratorio, las cuales ayudarán a familiarizar al usuario con simulaciones y sus herramientas. Además, se agregan otras experiencias, las que estarán relacionadas con el uso de características avanzadas de simulación. Dentro de este mismo bloque, se expone sobre la estructura del software en conjunto con conectividad exterior.

Finalmente, una vez conocido el software prácticamente en su totalidad, se realiza un análisis económico para su implementación.

1.5.1 Virtualización y requeriemientos de hardware

Debido a que Cisco VIRL utiliza recursos de virtualización para su funcionamiento [1], se hace necesario realizar un estudio al respecto para comprender la virtualización en sí. Esto permitirá escoger una herramienta de virtualización apropiada para el entorno en el cual se ejecutará el software. Por lo tanto, se estudiará qué hipervisores brindan soporte para VIRL y cuál de ellos es el más apropiado para utilizar en un PC tipo desktop, además de conocer los requerimientos de hardware de dicho PC.

1.5.2 Instalación

En esta sección, se ilustra una serie de pasos a seguir acompañados de imágenes, los cuales guiarán al usuario en la configuración y puesta en marcha del software.

Se comienza con la configuración de cinco redes virtuales, las que el software utilizará para el manejo de las simulaciones. A continuación se describe cómo instalar VIRL OVA para PC, se realiza la activación del software y su posterior validación. Por último, se instala la estación de trabajo VM Maestro.

1.5.3 Experiencias introductorias

Se realizarán una serie de ejercicios introductorios, básicos pero fundamentales, que tienen como objetivo familiarizar al usuario con el entorno de trabajo VM Maestro, en la creación de nuevas topologías con nodos IOSv, construir y visualizar configuraciones y por último correr simulaciones.

Estas experiencias estarán agrupadas como se indica a continuación:

- Introducción al entorno de trabajo VM Maestro.
- Creación de una topología nueva.
- Creación de una red básica con nodos IOSv.
- Construcción y visualización de configuraciones.
- Trabajo con simulaciones.

1.5.4 Experiencias avanzadas

Se realizará un conjunto de experiencias apoyadas en guías de laboratorio. Así, el usuario podrá familiarizarse con la manipulación del software. Estos ejercicios están considerados como avanzados, puesto que el enfoque es en trabajar con configuraciones de protocolos y enrutamiento, control de nodos y estados de interfaces, configuraciones de capa dos, entre otros.

Las experiencias estarán agrupadas en orden ascendente en su complejidad como se ilustra a continuación:

- Trabajo con configuraciones.
- Configuración de protocolos de enrutamiento.
- Control de nodos y estados de interfaz.
- Configuración de acceso al gestor de nodos.
- Configuración de conmutadores de capa 2.
- Utilización de nodos Específicos.

1.5.5 Características avanzadas

En esta sección se presentan temas relacionados a características avanzadas del software. Específicamente, se abordarán tres tópicos principales, estos son: APIs VIRL y automatización de funciones, herramientas avanzadas para utilizar en simulaciones como captura de paquetes, introducción de latencia, etc. y por último se expone sobre la estructura del software para concluir con conectividad exterior.

A continuación se despliega una lista con los tópicos que serán abordados:

- APIs VIRL y automatización de funciones.
- Captura de paquetes.
- Introducción de latencia, jitter y pérdida de paquetes.
- Inyección de rutas artificiales.
- Visual Traceroute.
- Generación de tráfico con Ostinato.
- Simulación test ancho de banda con iperf.
- Conectividad externa.

1.5.6 Análisis tecnico y económico

Finalmente, para concluir con el estudio, se realiza un análisis de las características habilitadas en las variaciones de las imágenes cisco IOSv, CSR1000v, IOS-XRv, NX-OSv y ASAv, para conocer

las diferencias que habría con equipos reales, además de la factibilidad en correr máquinas virtuales de otros propietarios. Por último, se analiza el escenario económico para la implementación de una topología con hardware real versus Cisco VIRL.

2 Virtualización y requerimientos de hardware

En este capítulo se dará a conocer la importancia y la manera en que opera la virtualización de recursos de hardware. Además se conocerá el hardware necesario para correr el software.

2.1 Virtualización

Virtualizar aporta ventajas y posibilidades únicas en la actualidad. Permite reducir costos en prácticamente todos los campos de actuación de la administración de sistemas. Brinda la posibilidad de centralizar y automatizar procesos [22].

Combinar tecnologías de virtualización o crear una infraestructura virtual, proporciona una capa de abstracción entre memoria RAM y CPU, almacenamiento (Disco duro) y hardware de red (NIC), y las aplicaciones que se ejecutan en el mismo, tal como se aprecia en la imagen a la derecha de la figura 2-1. El despliegue de la infraestructura virtual no es perjudicial, ya que las experiencias de los usuarios así lo indican [19].

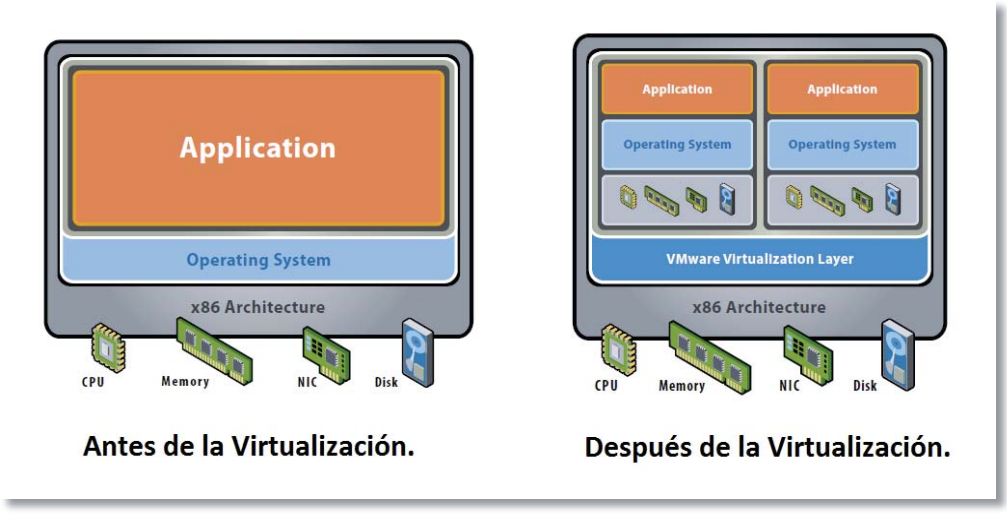


Figura 2-1. Virtualización. (Fuente: https://www.vmware.com/).

La figura 2-1 ilustra un concepto de máquina con arquitectura x86, antes de la virtualización y después de aplicar esta técnica. Pude apreciarse que los recursos de hardware disponibles son divididos para asignarlos a las diferentes máquinas virtuales soportadas sobre esta capa.

Existen dos enfoques para la virtualización en una máquina con arquitectura x86. Estos son, virtualización sobre un sistema operativo y la arquitectura de hipervisor (ver figura 2-2).

La virtualización sobre un sistema operativo estándar, ofrece servicios de partición en la parte superior de éste y es compatible con la más amplia variedad de configuraciones de hardware. En contraste con la arquitectura de un hipervisor, éste se aloja en el "fierro" de la máquina, es decir que no está sobre un sistema operativo (ver figura 2-2). Como un hipervisor tiene acceso directo a los recursos de hardware, es más eficiente que la virtualización sobre un sistema operativo, permitiendo una mayor escalabilidad, robustez y rendimiento [19].

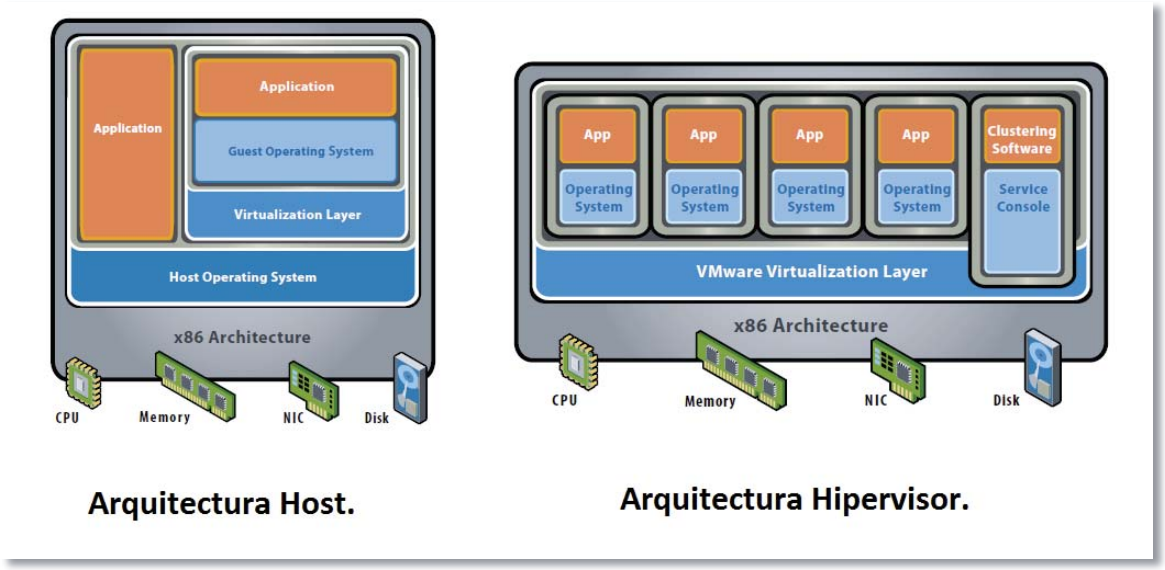


Figura 2-2. Arquitecturas de Host y Virtualización. (Fuente: https://www.vmware.com/).

La figura 2-2 ilustra dos enfoques distintos sobre virtualización. En la arquitectura de host, la virtualización es realizada sobre un sistema operativo a diferencia de la arquitectura de hipervisor donde la virtualización está sobre la máquina.

Considerando que se utilizará un PC tipo desktop para instalar el software, el cual cuenta con un sistema operativo anfitrión Windows 7 Profesional, es que se utilizará la arquitectura de host para correr el software.

2.2 Hipervisores soportados

VMware Inc., es una filial de EMC Corporation (propiedad a su vez de Dell Inc.) que proporciona software de virtualización disponible para ordenadores compatibles x86 [23].

Dentro de los productos que ofrece VMware, son 5 los hipervisores soportados que permiten correr VIRL [2]. Dentro de estas 5 opciones se encuentran disponibles hipervisores para MAC, Windows y Linux, de los cuales cada uno tiene sus diferencias de desempeño.

La lista siguiente indica los hipervisores soportados, la que es proporcionada por el sitio web oficial de VIRL [4].

1. VMware Fusion Pro v5.02 o superior (incluye v6.x o v7.x)
2. VMware Workstation para Windows o Linux v8.04 o superior (incluye v9 -> v12).
3. VMware Player v5.02 o superior (incluye v6.x y 7x).
4. VMware Workstation Player 12 o superior.
5. VMware ESXI 5.1/5.5/6.x utilizando cliente vSphere.

Cabe destacar que VIRL sólo funciona con hipervisores otorgados por VMware Inc [3].

De la lista anterior, se encontró que las distribuciones VMware Workstation (2.) y VMware Player (3.), ya no se encuentran disponibles en el sitio web oficial de VMware Inc. Sin embargo, VMware Workstation Pro viene a reemplazar VMware Workstation (2.), y VMware Workstation Player (anteriormente VMware Player Pro) reemplaza a VMware Player (3.).

Por lo tanto la lista de hipervisores soportados para correr VIRL se resume a la siguiente:

1. VMware Fusion Pro v5.02 o superior (incluye v6.x o v7.x).
2. VMware Workstation Pro 12.
3. VMware Workstation Player 12.
4. VMware ESXi 5.1/5.5/6.x utilizando cliente vSphere.

2.2.1 Qué hipervisor utilizar

En esta oportunidad, el enfoque será sobre dos hipervisores, VMware Workstation Pro y VMware Workstation Player. Puesto que para el propósito establecido, se utilizará un PC desktop, quedando descartado automáticamente VMware Fusion Pro ya que éste es utilizado en computadoras MAC. Por otra parte, VMware ESXi es un hipervisor que no necesita de un sistema operativo anfitrión para su funcionamiento, y su utilización es con fines avanzados. Por tanto, VMware ESXi también será descartado ya que se utilizará Windows 7 como sistema operativo anfitrión. Luego, la discriminación se reduce a VMware Workstation Pro y VMware Workstation Player.

Un antecedente importante y determinante a considerar, es que VIRL proporciona la capacidad de vincular las interfaces tanto de management-plane y data-plane para nodos dentro de la simulación a redes externas y dispositivos dentro de dichas redes. Esta conectividad requiere una interfaz compartida entre el PC host y la máquina virtual VIRL [4].

Tanto VMware Fusion Pro (MAC) y VMware Workstation Pro (Linux y Windows) proporcionan herramientas para configurar las redes virtuales e interfaces requeridas. VMware Workstation Player no incluyen este tipo de herramientas. Aunque VIRL seguirá funcionando con VMware

Workstation Player, no será posible la comunicación entre el PC host y los nodos dentro de la simulación. Si se desea tal conectividad, entonces será necesario utilizar VMware Fusion Pro para Mac o VMware Workstation Pro para Windows o Linux [4]. Por lo tanto, el hipervisor que se utilizará es WMware Workstation Pro.

El costo de una licencia nueva para WMware Workstation Pro, es de $187.49 USD [24], sin embargo, existe la versión de prueba, la que será suficiente para este estudio.

2.3 Requerimientos de hardware

A continuación se despliega una lista con los requerimientos mínimos de hardware para correr VIRL [4] [3]:

- El sistema principal debe ser capaz de acceder a Internet de forma regular (puertos TCP 4505 y 4506 de salida activados en cualquier firewall/proxy).
- Un mínimo de 8 GB de RAM y CPU de cuatro núcleos se debe asignar a la máquina virtual VIRL. Más recursos permite simulaciones de mayor tamaño; se sugiere 12GB por 20 nodos. Esto dependerá de los nodos iniciados en la simulación.
- Extensiones de virtualización VT-x/EPT o AMD-V/RVI para procesadores Intel y AMD respectivamente. Estas funciones deben estar habilitadas en la BIOS.
- 70 GB de espacio libre en disco para la instalación (se recomienda el uso de un SSD).

La distribución adquirida para este estudio, corresponde a la edición académica. La que puede funcionar con un máximo de 20 nodos Cisco. Sin embargo, la edición académica ya no se encuentra disponible. Sólo es posible adquirir la versión personal de 20 nodos, la que posee un costo de $199.9 USD por año [4].

Nota: Un procesador de doble núcleo y 8GB de RAM no podrá ejecutar VIRL, tal como lo indica [25].

El PC que se utilizará para instalar el software posee un procesador Intel Core i5 4440 CPU @3.10 [GHz], el que cuenta con tecnología de virtualización VT-x/EPT. Un total de 16 [GB] de memoria RAM y un HDD con suficiente espacio.

3 Instalación

En este capítulo se revisará todo lo necesario sobre la instalación de VIRL utilizando el hipervisor VMware Workstation Pro y Windows 7 como sistema operativo anfitrión.

Una vez verificado que el PC donde se instalará VIRL cumple con los requerimientos mínimos de hardware y contando con VMware Workstation Pro instalado, se procede como sigue:

3.1 Creación y configuración de redes virtuales

La máquina virtual VIRL requiere conexión con 5 redes virtuales únicas. La primera es para gestionar y es mapeada a 'VMnet8'/NAT por VMware Workstation por defecto. Esta asignación, provee a VIRL con una dirección IP, una puerta de enlace y un nombre de dominio para la dirección del servidor, y el acceso a Internet a través de la conexión de red del PC.

Las otras cuatro redes utilizadas por VIRL son para conexión externa para capa-2 y capa-3 ('Flat', 'Flat1', y 'SNAT') y clustering ('INT').

A continuación se despliegan los pasos a seguir para la configuración:

1. Abrir VMware Workstation Pro.
2. En menú seleccionar 'Edit'.
3. Abrir 'Virtual Network Editor'.
4. Seleccionar 'Add Network'.
5. Seleccione 'Ok' para utilizar la siguiente red virtual disponible ('VMnet1' por defecto).
6. Desactivar 'Use local DHCP...'
7. Colocar en el campo 'Subnet IP' la dirección IP: 172.16.1.0.
8. Mantener la máscara de subred por defecto '255.255.255.0'.
9. Seleccionar 'Ok'.
10. Seleccionar 'Apply' para crear una nueva red virtual.
11. Repetir pasos de 4 a 10 para configurar las otras 3 redes que necesita VIRL.
    - '172.16.2.0' para VMnet2
    - '172.16.3.0' para VMnet3
    - '172.16.10.0' para VMnet4.
12. Seleccionar 'Ok' para cerrar el editor de redes.

La figura 3-1, ilustra cómo debería quedar la configuración final de las 5 redes virtuales.

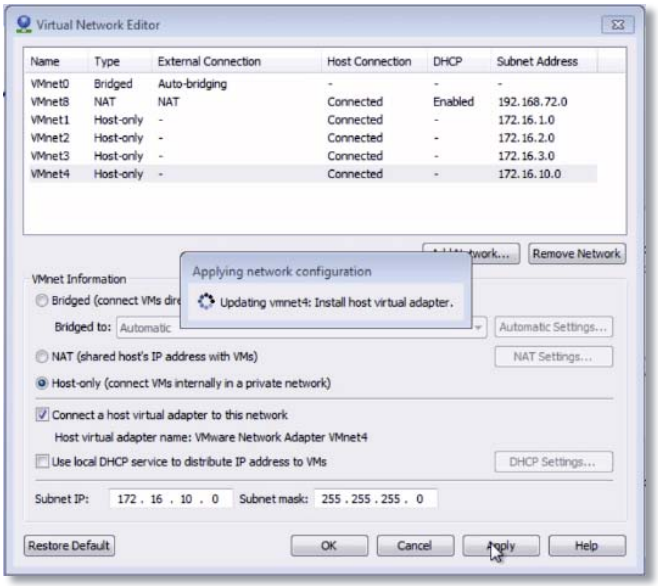


Figura 3-1. Configuración de redes virtuales. (Fuente: elaboración propia).

3.2 Implementación de VIRL OVA

VIRL OVA para PC, es la máquina virtual (VIRL) que debe ser integrada al hipervisor, y se le deben asignar los recursos necesarios para su funcionamiento. A continuación se muestran los pasos a seguir:

1. Abrir VMware Workstation Pro.
2. Buscar VIRL OVA en el lugar donde fue descargado.
3. Seleccionar 'Open a Virtual Machine'
4. Abrir VIRL OVA.
5. Editar el nombre si se desea. Seleccionar 'Import'.
6. Una vez importado, seleccionar 'Edit virtual machine settings'.
7. Seleccionar 'Network Adapter 2' y asignarlo a 'VMnet1'.
8. Seleccionar 'Network Adapter 3' y asignarlo a 'VMnet2'.
9. Seleccionar 'Network Adapter 4' y asignarlo a 'VMnet3'.
10. Seleccionar 'Network Adapter 5' y asignarlo a 'VMnet4'.
11. Seleccionar 'Processors'.
12. Adaptar el número de núcleos de procesador para adaptarse a las capacidades de hardware (4 núcleos).
13. Confirmar que 'Virtualize Intel VT-x/EPT...' esté habilitado.
14. Seleccionar 'Memory'.
15. Si se desea, adaptar la cantidad de memoria para adaptarse a las capacidades de hardware (8 GB como mínimo).
16. Seleccionar 'Ok' para terminar.

La figura 3-2, ilustra cómo debería quedar la configuración de la máquina virtual VIRL.

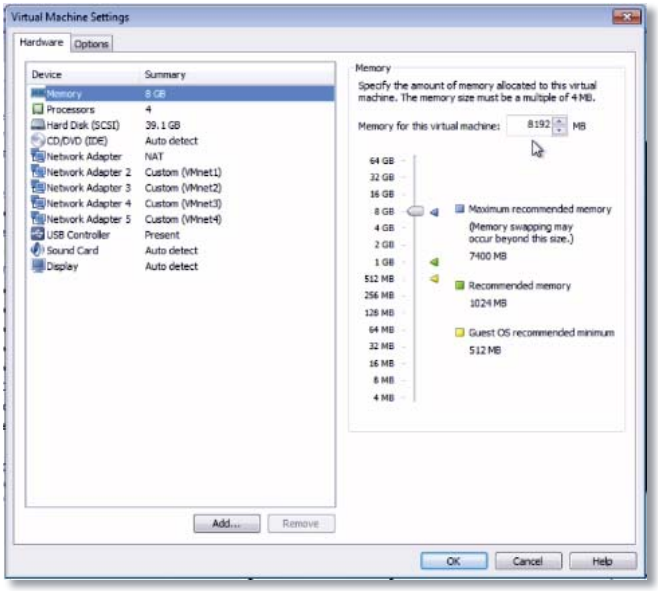


Figura 3-2. Configuración de la máquina virtual VIRL. (Fuente: elaboración propia).

3.3 Preparación activación

A continuación se presentan los pasos necesarios para preparar VIRL para la activación.

1. Correr la máquina virtual VIRL.
2. Colocar usuario “virl” y contraseña “VIRL”.
3. Doble clic en ‘xterm’ para abrir la terminal.
4. Maximizar la terminal para ver la configuración de manera correcta.
5. Introducir el comando ‘sudo kvm-ok’ y verifique la respuesta “KVM acceleration can be used”. Si la respuesta no es satisfactoria, se debe verificar que los pasos anteriores hayan sido correctos (3.1 y 3.2).
6. Utilizar uno de los siguientes comandos para verificar la conectividad a Internet.
    - ping -c 4 www.cisco.com
    - curl http://curltools.com/get-ip
    - wget --spider --no-verbose http://www.google.com
7. Examinar los resultados a los comandos anteriores: respuesta de ping, una IP address curl, o ‘200 OK’ para wget.
8. Cerrar la terminal con el comando exit.

La figura 3-3, ilustra la máquina virtual VIRL corriendo dentro del hipervisor. Se resalta en color amarillo la terminal ‘xterm’ utilizada en 3.3.

Figura 3-3. Vista de la máquina virtual VIRL. En amarillo se encuentra la terminal 'xterm'. (Fuente: elaboración propia).

3.4 Activación

Con el fin de que VIRL funcione correctamente, primero debe ser activado mediante el Sal-ID, Salt-Domain, y la clave RSA asociada con la licencia.

El procedimiento se describe a continuación.

1. Descargar y tener en cuenta la ubicación de 'VIRL license Salt-key', incluida en la compra.
2. Doble clic en el ícono 'IP Address' en el escritorio de VIRL VM.

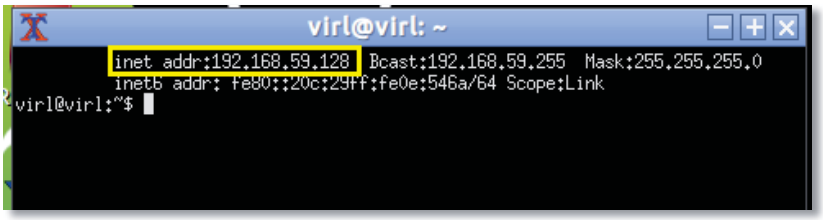


Figura 3-4. Dirección IP utilizada por VIRL en el laboratorio. (Fuente: elaboración propia).

3. Tomar nota de la dirección IP utilizada por VIRL (ver figura 3-4).
4. Cerrar la terminal.
5. Abrir un buscador web (en el PC host, no en la máquina virtual VIRL) e ir a la dirección IP obtenida en el punto anterior.

6. Entrar a 'User Workspace Management' con nombre de usuario 'uwmadmin' y password 'password'.
7. Seleccionar 'VIRL Server' en el menú que se encuentra a la izquierda.
8. Seleccionar 'Salt Configuration and Status' en el menú a la izquierda.
9. Seleccionar 'Reset Keys and ID' en la ventana principal.
10. Cortar y pegar el nombre del archivo license key (sin incluir '.pem') provisto en la compra, en el campo 'Salt ID and Domain'. (En este caso el nombre del archivo provisto por la compra es '1C39D176.virl.info').
11. Introducir la dirección de correo electrónico (con la cual se creó la cuenta para adquirir el software) en el campo 'Customer Email Address'.
12. Introducir dos o más nombres para VIRL salt servers. Por ejemplo:
    - us-1.virl.info, us-2.virl.info, us-3.virl.info, us-4.virl.info
    - eu-1.virl.info, eu-2.virl.info, eu-3.virl.info, eu-4.virl.info

    Nota: en la instalación se introdujo la configuración por defecto que sería la primera opción.

13. Eliminar el contenido existente del campo 'Minion private RSA key'.
14. Abrir el archivo license key en editor de texto (bloc de notas).
15. Seleccionar y copiar todo el contenido del archivo license key.
16. Cerrar editor de texto.
17. Pegar el contenido del archivo license key en el campo 'Minion private RSA key'.
18. Seleccione 'Reset' y esperar a que la página se actualice.
19. Localizar y seleccionar 'Check status now'.
20. Una vez que la página se actualiza, confirmar que la hora actual (esperar un momento) se encuentra en la lista 'Last successful contact'.
21. Cerrar el buscador web.

3.5 Validación de la Instalación

Con los siguientes pasos será posible validar que VIRL se ha instalado y configurado correctamente.

1. Doble clic en el ícono 'xterm' para abrir la consola (dentro de VIRL VM).
2. Visualizar el estado de los agentes OpenStack Neutron con el comando
   neutron agent-list
3. Verificar que por cada agente Neutron, en la columna '**alive**' se muestra '**:-)**'
   Debe haber un mínimo de 4 agentes Neutron presentes (podrían haber más), estos son:
   - Linux-bridge-agent
   - Metadata agent
   - DHCP agent
   - L3 agent
4. Comprobar que tanto los servicios de STD y UWM se encuentren activos y en estado 'listening' con el siguiente comando:

sudo virl_health_status | grep listening

5. Mostrar la configuración de licencia VIRL con el siguiente comando.

   sudo virl_health_status | grep -A 4 -e hostid -e product

6. Comprobar que los siguientes valores coinciden con los especificados en el contrato de licencia.
   - 'hostid' debe coincidir con salt-id y dominio indicado en el contrato de licencia.
   - 'product-capacity' número de nodos permitido por el contrato de licencia.
   - 'product-expires' debe ser '7'.

   'product-expires' indica el número máximo de días que VIRL permitirá correr simulaciones sin hacer contacto con los servidores Cisco SaltStack.

   La figura 3-5 ilustra la comprobación del punto 6.

Figura 3-5. Verificación de "hostid", "product-capacity" y "product-expires". (Fuente: elaboración propia).

7. Cerrar la terminal con exit.

3.6 Instalación y configuración de VM Maestro

VM Maestro es la aplicación del lado del usuario que se utiliza para construir topologías, generar configuraciones y visualizaciones, y gestionar simulaciones que se ejecutan.

VM Maestro viene en el paquete VIRL y está disponible para su instalación en Windows, OS X, y Linux. Los pasos necesarios para descargar y configurar VM Maestro se describen a continuación:

1. Abrir navegador web e ir a la dirección IP obtenida en 'IP Address'
2. Seleccionar 'VM Maestro Clients' en la lista de opciones.
3. Descargar VM Maestro client para Windows en este caso.
4. Instalar VM Maestro client.
5. Abrir VM Maestro (buscar el ícono) y reconocer las advertencias de seguridad que pueden aparecer.
6. Leer y reconocer el contrato de licencia.

7. Cuando se requiera la dirección del servidor, introduzca la dirección IP de VIRL VM recordada previamente (punto 2 de sección 3.4).
8. Seleccionar ‘Ok’ y utilizar el nombre de usuario por defecto ‘guest’ y la password ‘guest’.
9. Seleccionar el botón de credenciales (etiquetado como "guest") en la esquina inferior derecha de la ventana de la aplicación.
10. Confirmar que cada uno de los Servicios Web se indica como "Compatible" en color verde.
11. Seleccionar "Ok" para completar la configuración de la VM Maestro.

Se debe descargar todos los subtipos de nodos disponibles en VIRL a VM Maestro:

- Seleccionar en el menú “File”, luego “Preferences”, a continuación, “Node Subtypes”.
- Seleccionar “Restore Defaults”, luego “Ok”, luego “Apply”.
- Seleccionar “Fetch from Server”, luego “Ok”.
- Seleccionar “OK” para finalizar.

Se deben realizar estos pasos cada vez que se introduzca un nuevo subtipo de nodo.

La figura 3-6, es una captura de pantalla luego de descargar todos los subtipos de nodos disponibles en el servidor VIRL, los que fueron integrados a VM Maestro.

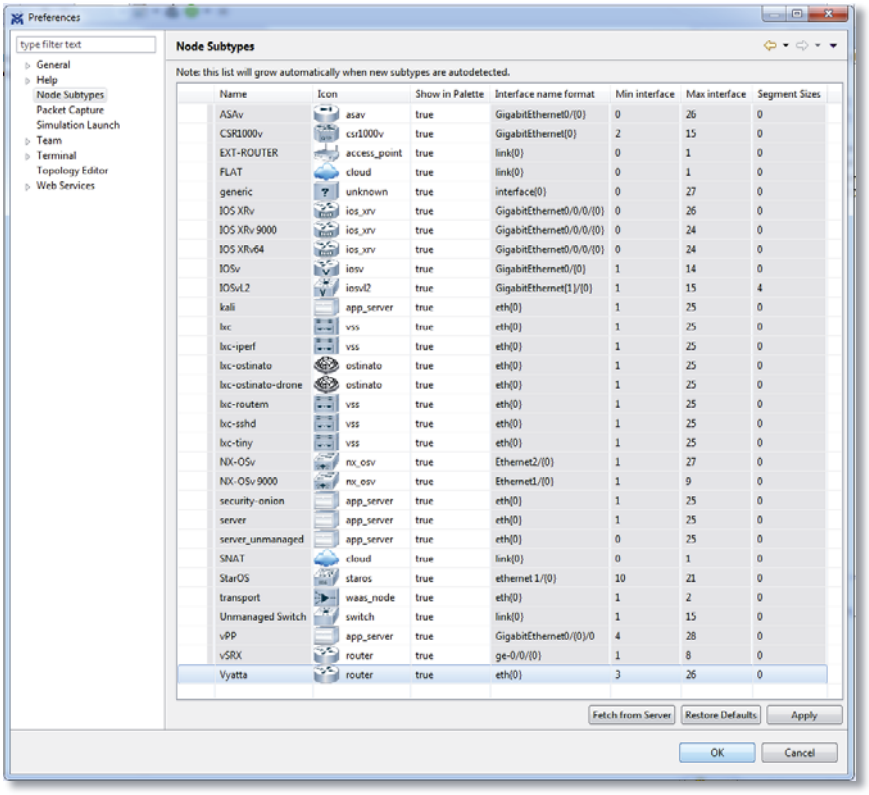


Figura 3-6. Subtipos de nodos disponibles en VIRL. (Fuente: elaboración propia).

Fin de la instalación.

4 Experiencias introductorias

A continuación, se realizarán una serie de ejercicios introductorios básicos pero fundamentales, que tienen como objetivo familiarizar al usuario con el entorno de trabajo VM Maestro, en la creación de nuevas topologías con nodos IOSv, construir y visualizar configuraciones, y por último correr simulaciones.

4.1 Introducción a VM Maestro

En este ejercicio se conocerá el diseño y la distribución (layout) de VM Maestro con el objetivo de acercarse y familiarizarse con las herramientas disponibles para el desarrollo de topologías con VIRL.

Para inicializar VM Maestro, ir al ícono que se ilustra en la figura 4-1, hacer doble clic y esperar a que se abra VM Maestro.

Figura 4-1. Ícono VM Maestro. (Fuente: elaboración propia).

VM Maestro está organizado en una serie de paneles o áreas, las que permitirán crear, definir y gestionar topologías. Estos son:

4.1.1 Panel editor de topologías

El panel editor de topologías es donde se crearán las topologías utilizando objetos y herramientas que se encuentran al lado izquierdo de dicho panel. Los objetos se pueden nombrar, mover o borrar desde el panel. También es posible acercar o alejar la topología al deslizar la rueda de desplazamiento del mouse o modificando el zoom en la barra de herramientas.

La figura 4-2, ilustra el panel editor de topologías resaltado en un cuadro amarillo.

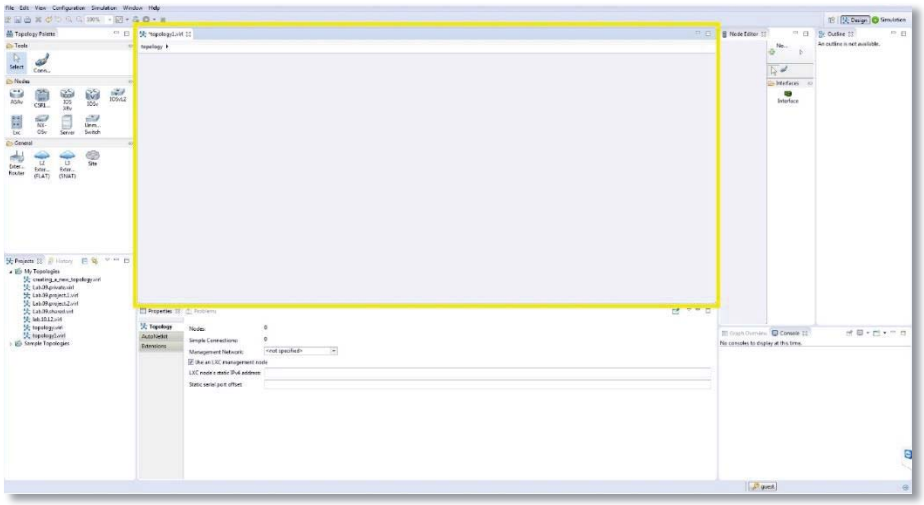


Figura 4-2. Panel Editor de Topología en VM Maestro. (Fuente: elaboración propia).

4.1.2 Panel de propiedades

El panel de propiedades proporciona un medio a través del cual las diversas opciones asociadas con los objetos en una topología o la propia topología pueden ser manipulados.

Dependiendo del objeto, varias pestañas aparecerán a la izquierda del panel de propiedades, las que proporcionan un medio para examinar los datos asociados al objeto. Podría pensarse que al hacer clic derecho en un objeto en el panel editor de topologías, se tendría acceso a sus propiedades pero esto no es así, el panel de propiedades es donde se visualizan y ajustan todas las opciones.

La figura 4-3, resalta en un cuadro amarillo el Panel de Propiedades.

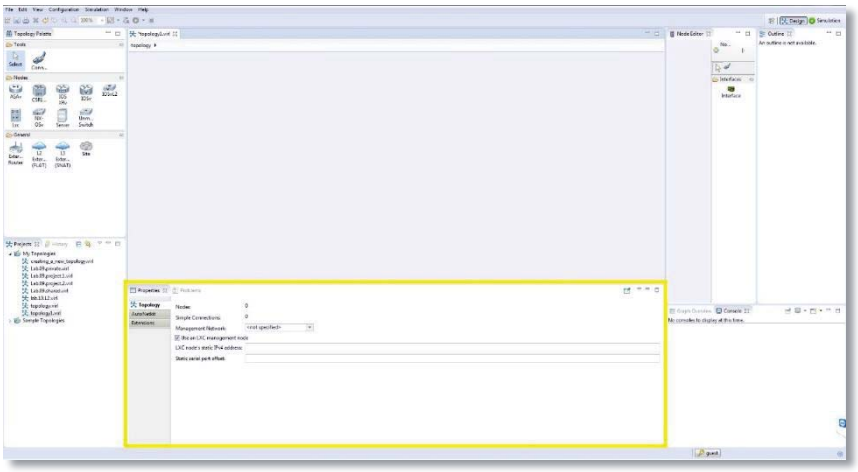


Figura 4-3. Panel de Propiedades en VM Maestro. (Fuente: elaboración propia).

4.1.3 Paleta de nodos y herramientas

En la paleta de nodos y herramientas se encuentran disponibles los objetos que serán utilizados pare realizar topologías.

La figura 4-4 resalta en un cuadro amarillo la paleta de nodos y herramientas.

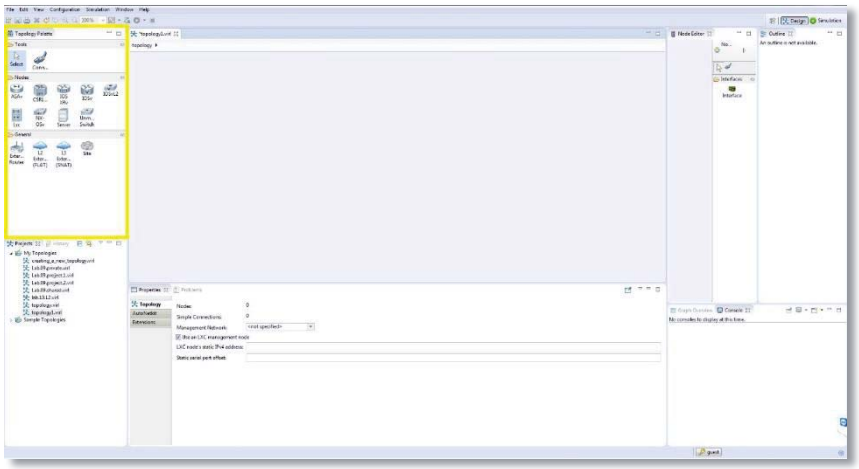


Figura 4-4. Paleta de Nodos y Herramientas en VM Maestro. (Fuente: elaboración propia).

Las herramientas y los objetos que serán utilizados en los siguientes ejercicios se muestran en la tabla 4-1.

Tabla 4-1. Herramientas de simulación.

Elemento	Descripción
A 	Herramienta de Selección: se utiliza para seleccionar objetos en el panel editor de nodos para ser manipulados.
B 	Guardar: se utilizar para guardar los cambios realizados en una topología.
C 	Herramienta Conectar: se utiliza para crear enlaces de red punto a punto en una topología.
D 	Router IOSv: se utiliza para crear uno o más routers IOSv en una topología.
E 	Switch IOSvL2: se utiliza para crear Switch IOSv de capa 2 en una topología.
F 	Router IOS-XRV: se utiliza para crear uno o más routers IOS-XRV en una topología.

G		Switch no Gestionable: se utiliza para crear un conmutador no administrado al que pueden conectarse otros nodos.
H		Herramienta de Conexión Externa L2: utilizado para crear una o más conexiones externas de capa 2 (Flat) o capa 3 (“SNAT”).
I		Nodo LXC: utilizado para crear uno o más contenedores Linux (LXC).
J		Nodo Servidor: utilizado para crear uno o más servidores Ubuntu LTS (u otros de terceros).

4.1.4 Panel de proyectos

El Panel de Proyectos proporciona un medio a través del cual se ilustran las topologías creadas, las que pueden ser administradas y eliminadas.

VM Maestro viene con 2 proyectos por defecto, llamados “My Topologies” y “Sample Topologies”.

La figura 4-5 resalta en un cuadro amarillo el Panel de Proyectos.

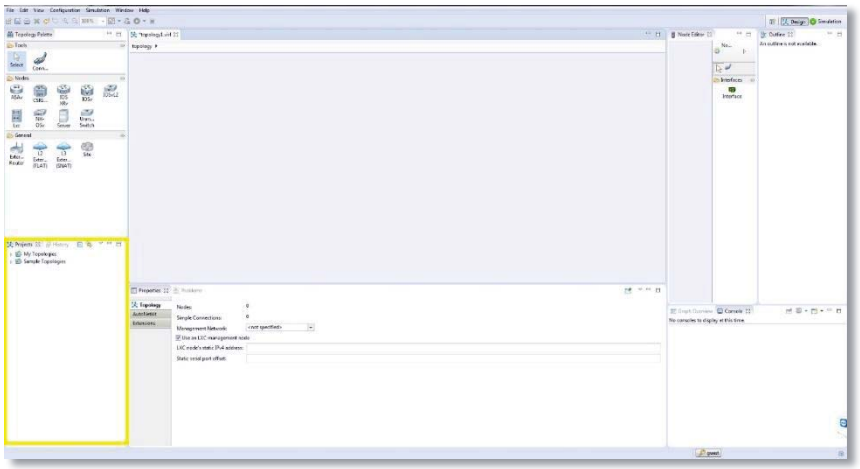


Figura 4-5. Panel de Proyectos en VM Maestro. (Fuente: elaboración propia).

4.1.5 Modos o perspectivas de VM Maestro

VM Maestro ofrece dos perspectivas o modos que definen el diseño de los diversos paneles. Estos son:

- Perspectiva de Diseño: ofrece una organización de paneles optimizados para el diseño de topologías, por medio de la inclusión de la Paleta de Nodos y Herramientas, además del Panel de Propiedades. Esta perspectiva está por defecto al abrir VM Maestro.
- Perspectiva de Simulación: ofrece una organización de paneles optimizados para trabajar con simulaciones, incluyendo espacio para mostrar las consolas de los Routers.

Estas perspectivas se seleccionan utilizando los dos botones denominados “Design” y “Simulation” en la esquina superior derecha de VM Maestro. En cualquiera de ambos modos los paneles pueden moverse, organizarse, quitar o añadir (a través de la opción “View” en el menú), para adaptarse a las preferencias de cada usuario.

Nota: si por alguna razón, alguna vez llegasen a faltar paneles importantes, es posible restablecer ambas perspectivas a su disposición por defecto haciendo clic derecho sobre la pestaña correspondiente y seleccionando “Reset”.

La figura 4-6 ilustra en un cuadro amarillo los botones utilizados para seleccionar ambas perspectivas o modos.

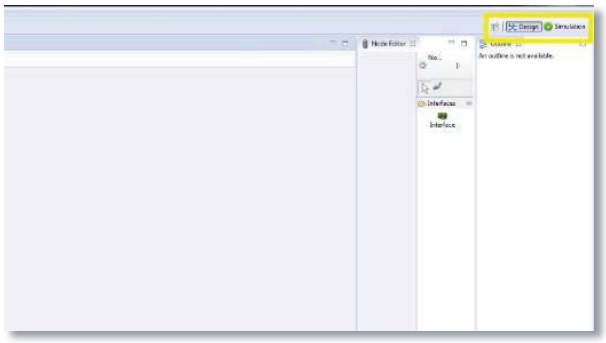


Figura 4-6. Botones para seleccionar la perspectiva de Diseño o Simulación en VM Maestro. (Fuente: elaboración propia).

4.1.6 Panel de simulaciones

Cuando se utiliza la perspectiva de Simulación, aparece un nuevo panel denominado **“Simulations”**.

La figura 4-7 ilustra en un cuadro amarillo el panel **“Simulations”**.

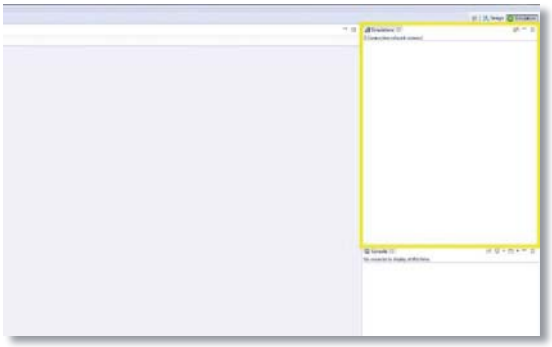


Figura 4-7. Panel de simulación en la perspectiva “Simulations”. (Fuente: elaboración propia).

Desde el panel de simulación, es posible manejar muchos aspectos de una simulación. Por ejemplo:

- Conectarse a las consolas de uno o más nodos.
- Detener simulaciones completas.
- Detener y reiniciar nodos de manera individual.
- Visualizar direcciones IP y puertos asociados con hosts y conectores.
- Extraer cambios en la configuración durante una simulación.
- Interactuar con simulaciones utilizando el “lienzo activo”.
- Ver visualizaciones en directo.

Cada uno de los aspectos que pueden manejarse en una simulación, serán vistos en detalles a medida que se avance en los ejercicios restantes.

4.1.7 Panel de consolas

En la parte de abajo del Panel de simulación puede apreciarse el Panel de Consolas.

La figura 4-8 resalta en un cuadro amarillo el Panel de Consolas.

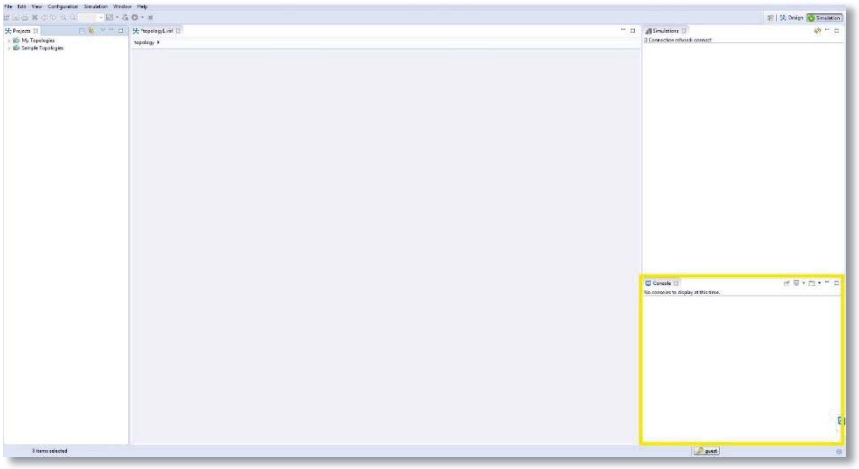


Figura 4-8. Panel de Consolas en la perspectiva de simulación. (Fuente: elaboración propia).

El Panel de Consolas proporciona una serie de pestañas que incluyen.

- Una consola para sistemas VIRL que proporcionan información acerca de las interacciones entre VM Maestro y la máquina virtual VIRL, así como información acerca de cada una de las medidas adoptadas para iniciar, detener y gestionar simulaciones.
- Consolas para los nodos que se ejecuten en una simulación. Siempre que sea solicitada una consola para un nodo, aparecerá una nueva pestaña en el panel de consolas.

Puede ser modificado el tamaño del Panel de Consolas o ser maximizado o minimizado utilizando los botones “Maximize” y “Minimize” respectivamente para adaptarse a las necesidades de espacio. El botón “Restore”, en la esquina del panel, restaurará el tamaño y posición original del Panel de Consolas.

4.1.8 Controles de topología y simulación

Por último, se tiene la barra de herramientas, la que se encuentra resaltada en un cuadro amarillo en la figura 4-9.

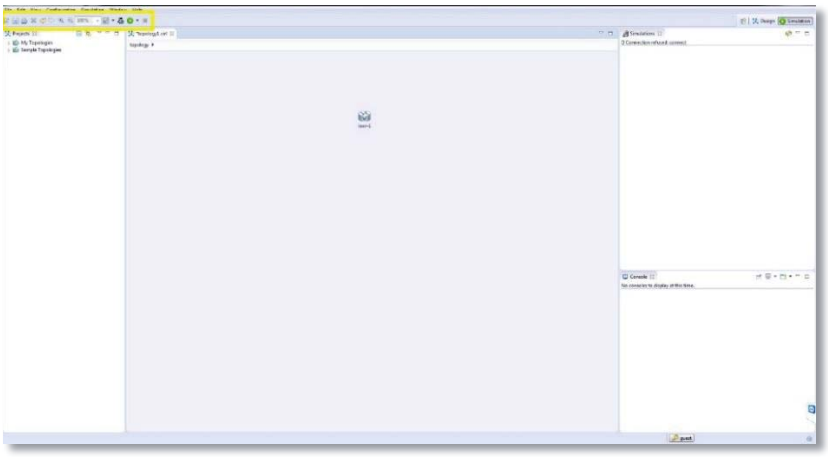


Figura 4-9. Controles de simulación y Topología en VM Maestro. (Fuente: elaboración propia).

Hay cuatro controles principales que se utilizan en los ejercicios siguientes. La tabla 4-2, muestra cada uno de estos controles, con una breve descripción de cada uno.

Tabla 4-2. Herramientas de simulación.

Elemento	Descripción
A 	Nueva Topología: se utiliza para crear un nuevo archivo de topología dentro de un proyecto existente.
B 	Construir configuraciones Iniciales: se utiliza para generar las configuraciones iniciales de los nodos de manera automática. Los nodos están configurados para recibir configuraciones generadas automáticamente por defecto. Este comportamiento puede ser desactivado.
C 	Correr Simulaciones: se utiliza para iniciar la simulación de una topología.
D 	Detener Simulación: se utiliza para detener una o más simulaciones en funcionamiento.

4.2 Creación de una nueva topología

En este ejercicios se creará una nueva topología y se configurarán las propiedades básicas que la afectan en su conjunto.

1. En el panel de Proyectos, seleccionar el proyecto llamado “My Topologies”.
2. Seleccionar “New Topology File” en la barra de herramientas (Elemento A de tabla 4-2).
3. Proporcionar un nombre a la nueva topología, asegurándose de añadir “.virl” al nombre.
4. Seleccionar “Finish” para crear el archivo para la topología.
5. Hacer clic en cualquier lugar en el panel de Topología para activar el panel de propiedades.
6. Seleccionar en el menú desplegable Management Network: “Private Simulation Network”.

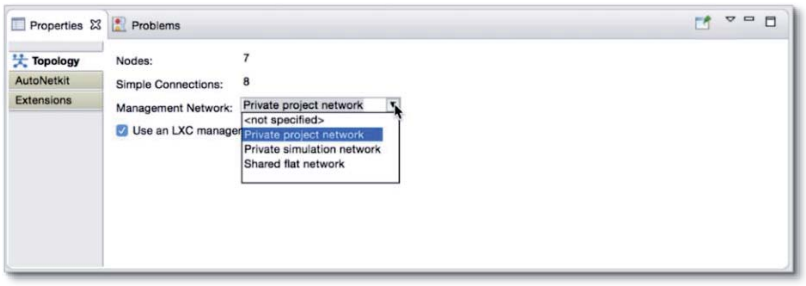


Figura 4-10. Selección de “Private Simulation Network”. (Fuente: elaboración propia).

La figura 4-10 muestra los tres modos de simulación: *Private project network, Private simulation network* y *Shared flat network*, donde es seleccionado el segundo modo.

7. Seleccionar la pestaña AutoNetkit en el panel de propiedades.
8. Maximizar el panel de Propiedades para visualizar todas las propiedades de AutoNetkit que pueden ser ajustadas.
9. Utilizar el menú desplegable para habilitar el uso de CDP en los Routers y Switches en la topología.

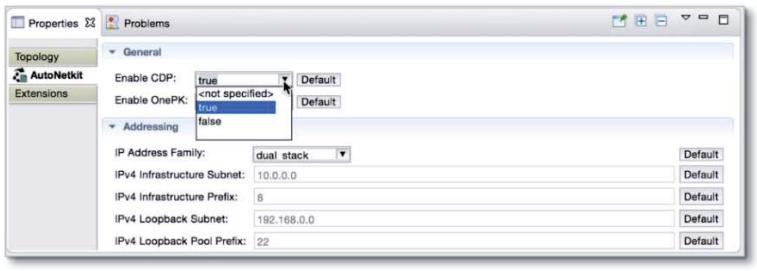


Figura 4-11. Habilitar el uso de CDP en los Router y Switch. (Fuente: elaboración propia).

La figura 4-11, ilustra la habilitación del uso de CDP (*Cisco Discovery Protocol*), en la pestaña AutoNetkit, tanto para Routers y Switches.

10. Utilizar el menú desplegable “IP Address family” para habilitar **dual_stack** en la topología.

Figura 4-12. Habilitar dual_stack en la topología. (Fuente: elaboración propia).

La figura 4-12, ilustra la habilitación de las versiones de direcciones IP en la pestaña AutoNetkit, donde son habilitadas las versiones 4 y 6, seleccionado la opción dual_stack.

11. Revisar las diversas propiedades de adaptación que están disponibles para dirección IP y observar que cada una fue asignada con un valor predeterminado.

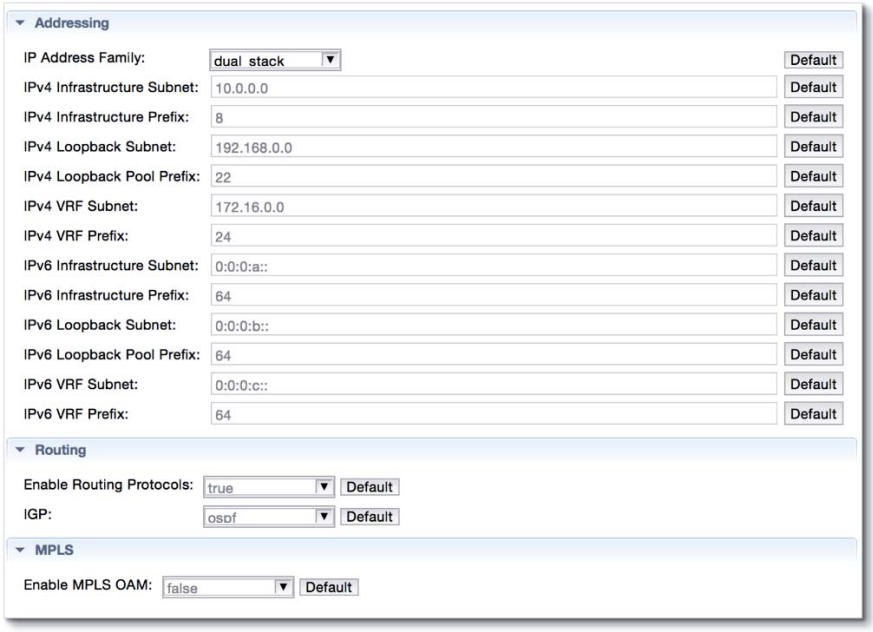


Figura 4-13. Propiedades de dirección IP. (Fuente: elaboración propia).

La figura 4-13, ilustra los parámetros con los que se realizarán las configuraciones tanto para el direccionamiento IP, protocolos de enrutamiento y MPLS (*Multiprotocol Label Switching*). En este caso, MPLS no está habilitado.

12. Observar que la configuración de los protocolos de enrutamiento, específicamente el uso de OSPF, está activado por defecto.
13. Restaurar el panel de Propiedades a su tamaño y posición original.

4.3 Creación de una red básica con nodos IOSv

En este ejercicio se creará y conectará una red simple utilizando nodos IOSv. Esta red será utilizada como base para varios de los ejercicios que continúan.

1. Seleccionar en la paleta de herramientas el nodo IOSv. (Elemento D de la tabla 4-1)
2. Dejar una serie de Routers en el panel de topología como se muestra en la figura 4-14.

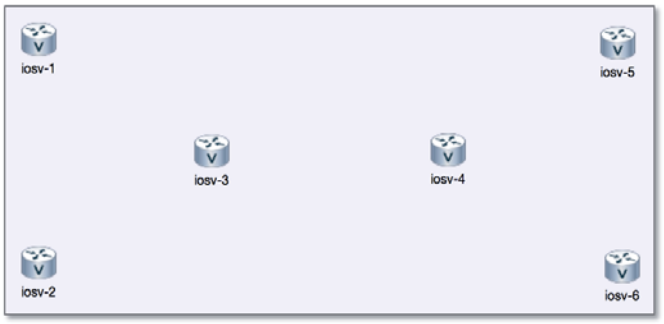


Figura 4-14. Distribución de nodos IOSv. (Fuente: elaboración propia).

La figura 4-14, ilustra la distribución de 6 nodos IOSv en el panel editor de topologías.

Cuando se utilizan nodos u otras herramientas de objetos, sólo se debe seleccionar la herramienta con un clic y luego "soltar" (hacer clic), tantas veces como copias sean requeridas en el panel de topologías. Al pulsar "Escape" se libera la herramienta seleccionada. Tratando de arrastrar y soltar no se conseguirá colocar ninguna herramienta.

3. Hacer clic en el nodo iosv-6, esperar un instante y realizar otro clic, para poder cambiar el nombre.
4. Nombrar al nodo como "Paris" y presionar "Enter" para aceptar la edición.
   El nombre también podrá ser modificado en el panel de propiedades (pestaña "Node") al seleccionar un nodo.
5. Cambiar el nombre a los demás nodos como se muestra en la figura 4-15.

Figura 4-15. Asignación de nombres a nodos IOSv. (Fuente: elaboración propia).

6. Seleccionar la herramienta Conectar en la Paleta de Herramientas. (Elemento C de la tabla 4-1).
7. Hacer clic en el nodo “Chicago” para seleccionarlo como punto de partida en la conexión.
8. Realizar la conexión con el nodo “Dallas”.
9. De la misma manera, conectar los nodos restantes como se muestra en la figura 4-16.

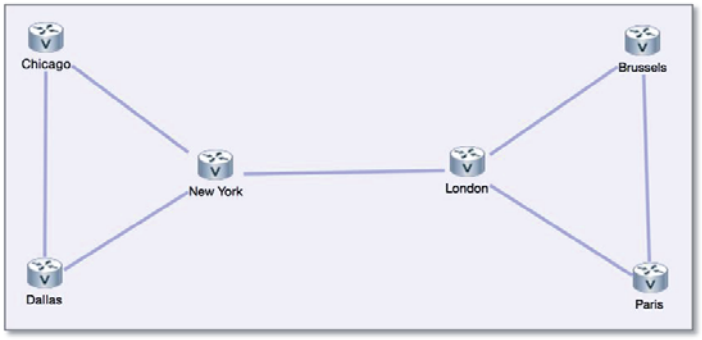


Figura 4-16. Conexión entre nodos IOSv. (Fuente: elaboración propia).

Hasta el momento, se posee una topología simple de una red definida, sin embargo, ninguno de los routers están configurados. En el siguiente ejercicio, se verá cómo generar configuraciones de manera automática y utilizar visualizaciones para explorar los detalles de la red.

4.4 Construcción y visualización de configuraciones

En este ejercicio se utilizará la herramienta AutoNetkit para generar configuraciones en cada uno de los routers, además se examinarán las visualizaciones que describen aspectos claves en la topología.

Hasta el momento se han definido los tipos de Routers, la cantidad de estos y sus conexiones, pero no mucho más. Por ahora, un conjunto de valores serán utilizados por AutoNetkit para generar configuraciones. Más adelante se verá cómo ajustar estas.

1. Seleccionar “Build Initial Configurations” en la barra de herramientas. (Elemento B de la tabla 4-2).
2. Aparecerá un mensaje preguntando si se desea ver los cambios en la configuración de los Routers, seleccionar “Yes”.

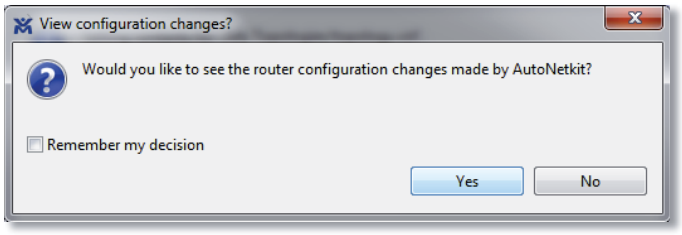


Figura 4-17. Mensaje para revisar los cambios en las configuraciones. (Fuente: elaboración propia).

El mensaje emergente mostrado en la figura 4-17, permitirá (o no) visualizar un paralelo entre la configuración previa y la actual de los diferentes nodos en la topología, tal como se aprecia en la figura 4-18.

3. Ver las configuraciones que se generaron para cada nodo en la topología.
   Notar que el costado Izquierdo de la figura 4-18 tiene 6 cuadros. Estos aluden a cada uno de los Routers. Además, se encuentra seleccionado el primero, que corresponde al nodo "Chicago".

   La ventana de configuraciones que ilustra la figura 4-18, ofrece una vista completa del archivo de definición de topología en XML, antes y después de utilizar la herramienta AutoNetKit. A la izquierda aparece la configuración antes de AutoNetKit y a la derecha después de la configuración. Dado que antes no existían configuraciones, se ve la totalidad de la configuración para cada Router. Posteriormente, sólo se mostraran cambios discretos cuando exista una nueva configuración.

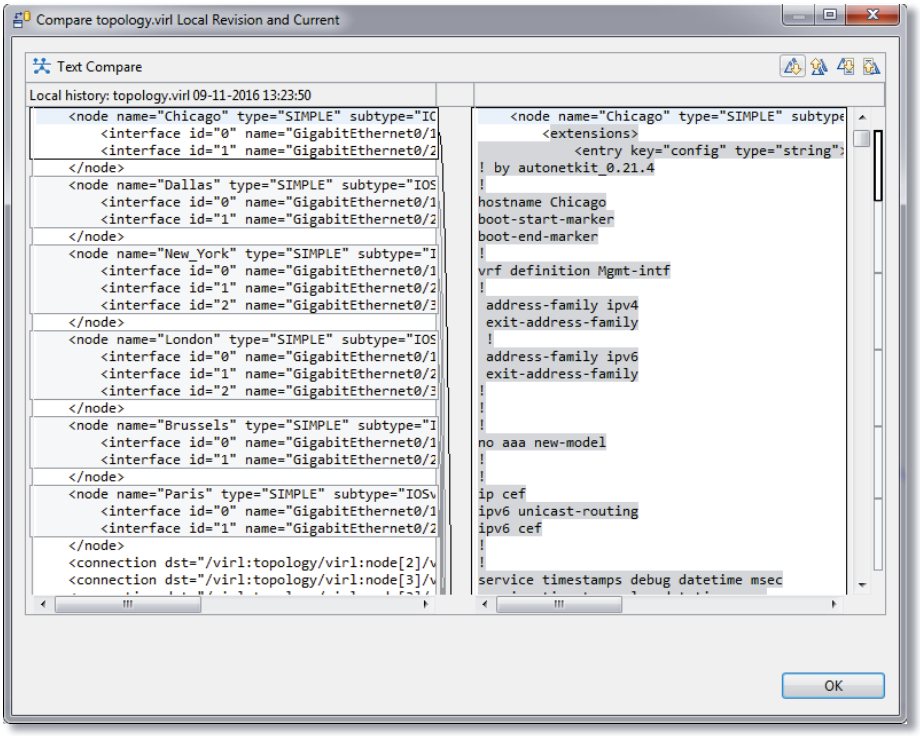


Figura 4-18. Configuraciones de los Routers por medio de AutoNetKit. (Fuente: elaboración propia).

4. Seleccione "OK" para cerrar la ventana de comparación.
5. Posteriormente aparecerá un mensaje, como el que se aprecia en la figura 4-19, preguntando si se desea ver las visualizaciones AutoNetkit, seleccionar "Yes". Las visualizaciones aparecerán automáticamente en una ventana del navegador Web.

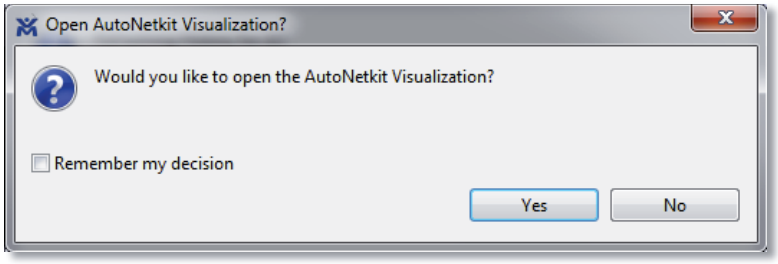


Figura 4-19. Mensaje para abrir la visualización AutoNetkit. (Fuente: elaboración propia).

6. Utilizar el selector de vista en la esquina superior izquierda para seleccionar entre las visualizaciones creadas por AutoNetkit. En el costado izquierdo de la figura 4-20 se aprecian las variadas visualizaciones disponibles.

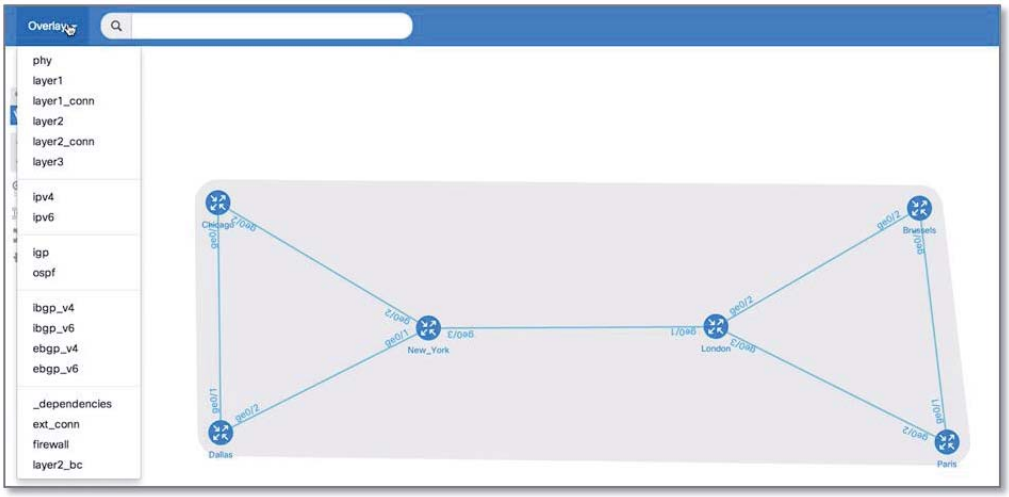


Figura 4-20. Visualizaciones de AutoNetkit. (Fuente: elaboración propia).

Seleccionar “ipv4” para poder apreciar las configuraciones de cada nodo con detalle. Al pasar por sobre los nodos y enlaces, aparecen los detalles acerca de dichos objetos. La figura 4-21, muestra el detalle del nodo “Chicago”, donde la dirección IP asignada a este es 192.168.0.6. Notar que en la figura 4-13, la dirección IPv4 de la subred corresponde a 192.168.0.0, donde las direcciones IP asignadas a los Routers van desde 192.168.0.1 hasta 192.168.0.6 respectivamente.

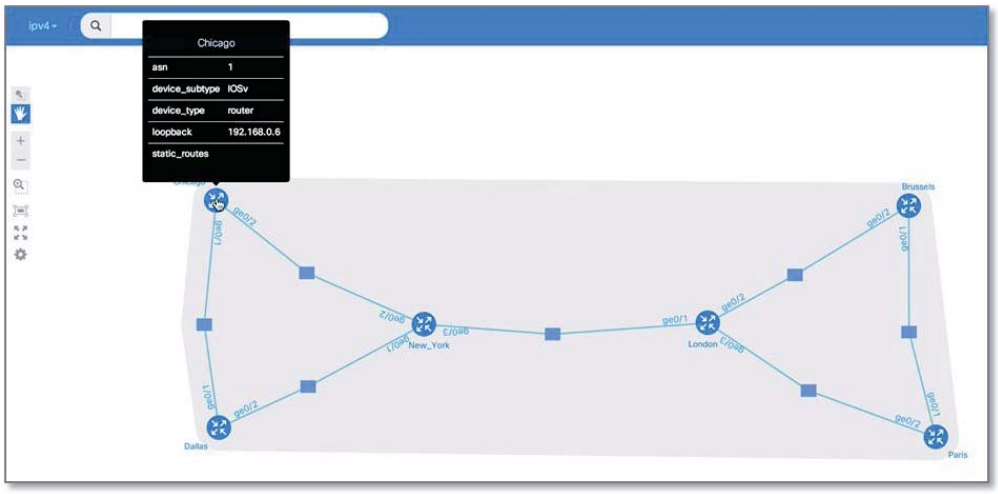


Figura 4-21. Visualización IPv4 AutoNetkit. (Fuente: elaboración propia).

La topología ilustrada por la figura 4-21 no es muy compleja, por tanto las visualizaciones podrían resultar un tanto "aburridas". Sin embargo, así como las topologías se vuelven más complejas, las visualizaciones llegan a ser extremadamente útiles para garantizar que la red esté configurada de la manera en que se pretendía, además de detectar problemas de configuración.

7. Una vez terminada la exploración por las visualizaciones, cerrar la ventana del navegador Web y retornar a VM Maestro.
8. Observar que luego de realizar las configuraciones a los nodos, aparecen etiquetas que incluyen las direcciones IPv4 e IPv6.

4.5 Trabajando con simulaciones

En este ejercicio se pondrá en marcha la simulación de la topología antes creada, se hará la conexión con la consola de uno de los routers simulados, y además, se confirmará la conectividad y el funcionamiento de los diversos protocolos de red.

1. Presionar el botón de lanzamiento de simulación en la barra de herramientas. (Elemento C de la tabla 4-2).
2. Dependiendo de la carga del servidor, se puede presentar un mensaje como "launching Simulation". Si este es el caso, seleccionar "Run in Background".
3. Después que la simulación fue lanzada, se presentará el mensaje "Simulation Launched", seleccionar "OK".
4. VM Maestro cambiará a la perspectiva de simulación. Esperar y observar que cada uno de los nodos en la simulación aparezcan como "[ACTIVE]", como se muestra en la figura 4-22.

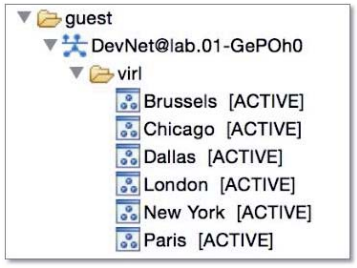


Figura 4-22. Nodos activos después de lanzar la simulación. (Fuente: elaboración propia).

5. Observar que al lanzar la simulación se abre una nueva pestaña en el lienzo, este es el "paño activo", el cual muestra el estado actual de cada nodo utilizando un color. Por ejemplo:
    - El color azul indica que el nodo está iniciando.
    - El color verde indica que el nodo está activo.
    - El color amarillo indica que el nodo está finalizando.
    - Sin color indica que el nodo está inactivo.

    La figura 4-23 ilustra el "paño activo".

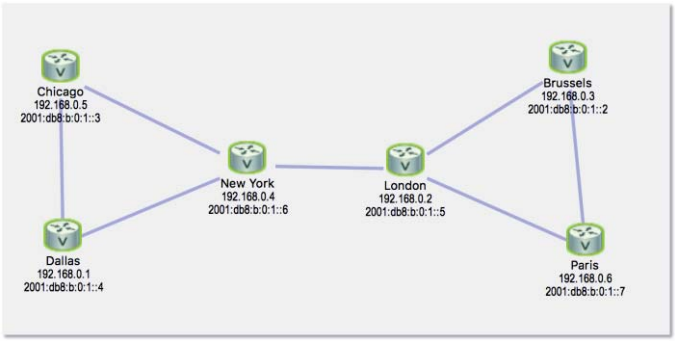


Figura 4-23. Colores correspondientes a los estados de cada nodo y conexión. (Fuente: elaboración propia).

El estado de los enlaces, son mostrados según:

- Las líneas discontinuas indican que el enlace está abajo.
- Las líneas continuas indican que el enlace está arriba.
- Decoraciones en los extremos de los enlaces indican razones conocidas por las que la conexión está abajo.

Las interfaces y los estados de las interfaces serán abordados con posterioridad.

6. Hacer clic en el nodo "London" y seleccionar "Telnet".
7. Desde el submenú seleccionar "to its Console port". La figura 4-24 ilustra lo anterior.

Figura 4-24. Acceso al puerto consola del nodo “London”. (Fuente: elaboración propia).

8. Maximizar el panel de la consola que aparece en la esquina inferior derecha.
9. Observar el proceso de arranque del nodo, hasta que aparezcan las adyacencias BGP, como se muestra en la figura 4-25.

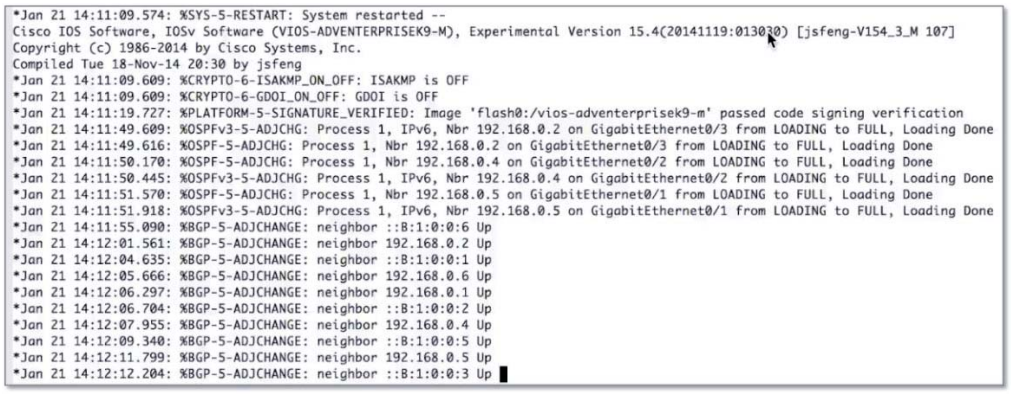


Figura 4-25. Visualización de la consola en la Iniciación del nodo London. (Fuente: elaboración propia).

10. Presionar “Enter” para obtener un indicador en la consola.
11. Activar “Enable Mode”, escribiendo “ena” y pulsando “Enter”. La contraseña es “cisco”.
12. Confirmar que los otros nodos están presentes y se han creado adyacencias CDP.

    La figura 4-26 ilustra el comando y la respuesta.

```
London#show cdp neighbor
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater, P - Phone,
                  D - Remote, C - CVTA, M - Two-port Mac Relay

Device ID        Local Intrfce     Holdtme    Capability  Platform  Port ID
New_York         Gig 0/1           139              R B   IOSv      Gig 0/3
New_York         Gig 0/0           146              R B   IOSv      Gig 0/0
Dallas           Gig 0/0           148              R B   IOSv      Gig 0/0
Brussels         Gig 0/2           138              R B   IOSv      Gig 0/2
Brussels         Gig 0/0           137              R B   IOSv      Gig 0/0
Chicago          Gig 0/0           134              R B   IOSv      Gig 0/0
Paris            Gig 0/3           148              R B   IOSv      Gig 0/2
Paris            Gig 0/0           144              R B   IOSv      Gig 0/0

Total cdp entries displayed : 8
```

Figura 4-26. Adyacencias CDP del nodo “London”. (Fuente: elaboración propia).

13. Confirmar que se han creado adyacencias OSPF. La figura 4-27 muestra el comando y la respuesta.

```
London#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
192.168.0.2       1   FULL/BDR        00:00:39    10.0.0.14       GigabitEthernet0/3
192.168.0.4       1   FULL/DR         00:00:37    10.0.0.25       GigabitEthernet0/2
192.168.0.5       1   FULL/DR         00:00:36    10.0.0.10       GigabitEthernet0/1
```

Figura 4-27. Adyacencias OSPF en el nodo “London”. (Fuente: elaboración propia).

14. Confirmar que se han establecido relaciones BGP. La figura 4-28 ilustra la respuesta al comando “show ip bgp summary”.

```
BGP router identifier 192.168.0.3, local AS number 1
BGP table version is 14, main routing table version 14
6 network entries using 840 bytes of memory
6 path entries using 480 bytes of memory
2/2 BGP path/bestpath attribute entries using 304 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 1624 total bytes of memory
BGP activity 7/0 prefixes, 7/0 paths, scan interval 60 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.168.0.1     4            1       8       9       14    0    0 00:00:58        1
192.168.0.2     4            1       7       7       14    0    0 00:01:02        1
192.168.0.4     4            1       9       8       14    0    0 00:00:56        1
192.168.0.5     4            1       8       8       14    0    0 00:00:52        1
192.168.0.6     4            1       8       9       14    0    0 00:00:58        1
```

Figura 4-28. Relaciones BGP del nodo “London”. (Fuente: elaboración propia).

15. Mostrar la tabla de enrutamiento IP y observar que las rutas han sido aprendidas a través de protocolo OSPF. La figura 4-29 ilustra esto por medio del comando “show ip route”.

```
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is not set

      10.0.0.0/8 is variably subnetted, 12 subnets, 3 masks
O        10.0.0.4/30 [110/2] via 10.0.0.25, 00:01:08, GigabitEthernet0/2
                     [110/2] via 10.0.0.14, 00:01:08, GigabitEthernet0/3
C        10.0.0.8/30 is directly connected, GigabitEthernet0/1
L        10.0.0.9/32 is directly connected, GigabitEthernet0/1
C        10.0.0.12/30 is directly connected, GigabitEthernet0/3
L        10.0.0.13/32 is directly connected, GigabitEthernet0/3
O        10.0.0.16/30 [110/2] via 10.0.0.10, 00:01:08, GigabitEthernet0/1
O        10.0.0.20/30 [110/3] via 10.0.0.10, 00:01:08, GigabitEthernet0/1
C        10.0.0.24/30 is directly connected, GigabitEthernet0/2
L        10.0.0.26/32 is directly connected, GigabitEthernet0/2
O        10.0.0.28/30 [110/2] via 10.0.0.10, 00:01:08, GigabitEthernet0/1
C        10.255.0.0/16 is directly connected, GigabitEthernet0/0
L        10.255.0.4/32 is directly connected, GigabitEthernet0/0
      192.168.0.0/32 is subnetted, 6 subnets
O        192.168.0.1 [110/3] via 10.0.0.10, 00:01:08, GigabitEthernet0/1
O        192.168.0.2 [110/2] via 10.0.0.14, 00:01:18, GigabitEthernet0/3
C        192.168.0.3 is directly connected, Loopback0
O        192.168.0.4 [110/2] via 10.0.0.25, 00:01:08, GigabitEthernet0/2
O        192.168.0.5 [110/2] via 10.0.0.10, 00:01:08, GigabitEthernet0/1
O        192.168.0.6 [110/3] via 10.0.0.10, 00:01:08, GigabitEthernet0/1
```

Figura 4-29. Tabla de enrutamiento IP. (Fuente: elaboración propia).

16. Seleccionar una dirección IP de la tabla de enrutamiento y realizar un ping para confirmar la accesibilidad. En este caso el ping se realiza a 192.168.0.6. La figura 4-30 muestra la respuesta exitosa.

```
Sending 5, 100-byte ICMP Echos to 192.168.0.6, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 2/7/14 ms
```

Figura 4-30. Respuesta exitosa de ping a 192.168.0.6. (Fuente: elaboración propia).

17. Salir de la consola con comando “exit”.
18. Restaurar el tamaño y posición de la consola.
19. Hacer clic en el identificador de simulaciones y seleccionar “Stop simulation”.

La figura 4-31, ilustra el identificador de simulaciones, donde se selecciona la opción para detener la simulación. Observar que dentro de dicha simulación se encuentran todos los nodos, los cuales serán detenidos uno por uno.

Figura 4-31. Detención de la simulación. (Fuente: elaboración propia).

20. Aparecerá una venta para confirmar la detención de la simulación. Marcar la casilla "Close associated terminal views.", y luego seleccionar "OK".
21. Volver a la perspectiva "Design".

Resumen

Luego de conocer la estación de trabajo VM Maestro, se ha construido una topología, se configuró, fue visualizada con la herramienta AutoNetkit, se corrió la simulación y se confirmó que la topología se comporta como una red verdadera.

Asegurarse de estar cómodo y manejar estas nuevas habilidades, puesto que lo visto hasta ahora será omitido en ejercicios posteriores. Es fundamental tener claridad respecto de lo que se ha hecho hasta el momento para evitar confusiones y poder avanzar sin dificultades en las capacidades de Cisco VIRL.

5 Experiencias avanzadas

En este capítulo, se realizarán una serie de ejercicios guiados por fichas de laboratorio, las que tienen como objetivo trabajar con configuraciones de protocolos y enrutamiento, control de nodos y estados de interfaces, configuraciones de capa dos y la utilización de nodos específicos. Estos ejercicios están considerados como avanzados, a diferencia de las experiencias expuestas en el capítulo 4, donde se pretende familiarizar al usuario con cosas básicas de manipulación.

El hecho de que se denomine Ejercicios Avanzados a este capítulo, no quiere decir que sea sinónimo de complejidad o dificultad, las actividades a realizar están detalladas para no incurrir en dudas que puedan ocasionar problemas.

5.1 Trabajo con configuraciones

Siempre que una simulación se ha puesto en marcha, la configuración contenida en la pestaña Configuración del Panel de Propiedades de cada nodo, se traspasa a cada Router durante el inicio [26].

Las configuraciones se pueden generar y manipular de varias maneras, incluyendo:

- Generada utilizando AutoNetkit (automáticamente).
- Generada utilizando AutoNetkit y modificarla.
- Generada a mano.
- Pegado de un único texto fuente.
- Generada o modificada directamente de un nodo en ejecución y luego extraer la configuración.
- Exportadas para su edición o para utilizarlo en “nodos reales”, en redes reales.

Explorar todos estos métodos se tornaría demasiado extenso, sin embargo se abordará:

- Añadir un nuevo nodo a la topología existente.
- Generar nuevas configuraciones, primero omitiendo el nuevo nodo y luego incluyéndolo.
- Modificar la configuración del nodo en VM Maestro.
- Modificar la configuración actual del nodo.
- Extraer y guardar la configuración modificada.

5.1.1 Experiencia N°1

Título: Trabajo con configuraciones.

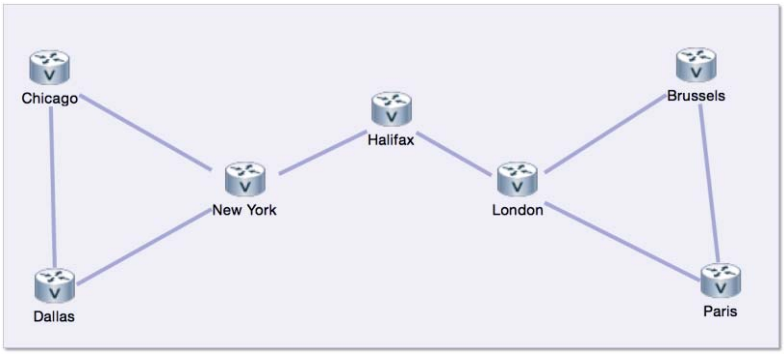


Figura 5-1. Topología a implementar. (Fuente: elaboración propia).

Objetivo:

El objetivo de esta experiencia es lograr que el usuario se familiarice con las variadas posibilidades de realizar configuraciones en nodos IOSv (routers) de la topología ilustrada en la figura 5-1.

Nota: la consonante “v” que acompaña a “IOS” se refiere a virtualización. Es decir que es un nodo IOS, o sistema operativo IOS virtualizable.

Equipamiento:

- 7 routers IOSv

Desarrollo:

Paso N° 1.

Construir la topología ilustrada por la figura 5-1. Seleccionar el nodo “Halifax”, y luego en el panel de propiedades, desactivar la opción “Auto-generate the Configuration based on these attributes.”, tal como se muestra en la figura 5-2.

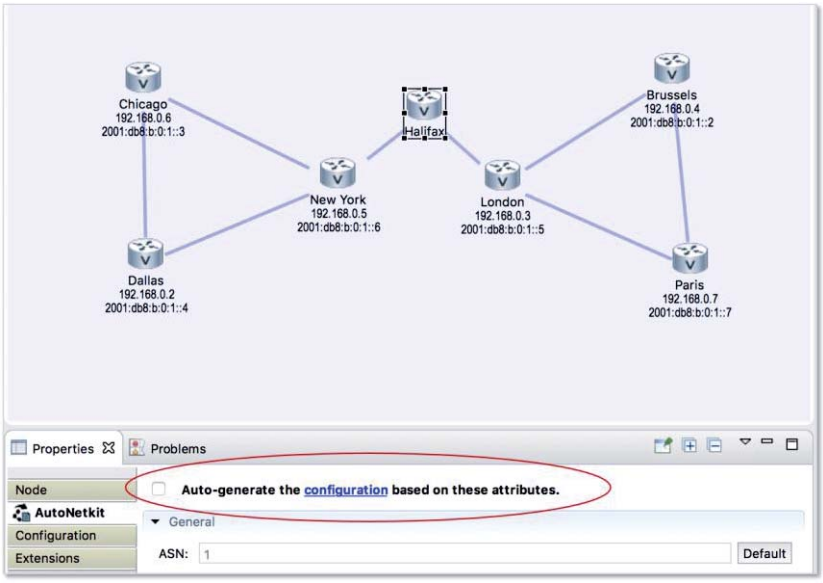


Figura 5-2. Desactivación de generación automática de configuración del nodo “Halifax”. (Fuente: elaboración propia).

Guardar los cambios de la topología (elemento B de la tabla 4-1). Generar las nuevas configuraciones (elemento B de la tabla 4-2) y examinar la comparación entre las configuraciones. Luego, examinar la pestaña configuración para el nodo “Halifax” y observar que no hay configuración, tal como se esperaba. Ver figura 5-3.

Figura 5-3. Visualización de la no configuración del nodo “Halifax”. (Fuente: elaboración propia).

Seleccionar la pestaña AutoNetkit y volver a habilitar la generación automática de configuración para el nodo “Halifax” (ver figura 5-2). Volver a generar configuraciones para la topología (elemento B de la tabla 4-2). En esta ocasión, el nodo “Halifax” ha sido incluido. Examinar la comparación entre las configuraciones y examinar la pestaña configuración del nodo “Halifax”. Observar que ahora existe una configuración para dicho nodo.

La herramienta AutoNetkit, está realizando la mayor parte del trabajo pesado al generar las configuraciones automáticas. Podría realizarse la configuración manual del nodo “Halifax”; si este es el caso, se debe tener en cuenta:

- Todas las interfaces deben ser Gigabit Ethernet.
- El número de la interfaz debe comenzar en cero y luego incrementar secuencialmente.
- La primera interfaz (0/0), siempre se reserva para la gestión del nodo.

Paso N° 2.

Examinar la configuración del nodo “Halifax”, específicamente por debajo de la interfaz “loopback0”. Cambiar la configuración (dentro de la pestaña configuración), y crear una nueva interfaz llamada “loopback10”. Proporcionar la dirección IP 10.10.8.1, con la máscara de subred 255.255.255.255 a la nueva interfaz “loopback10” (ver figura 5-4).

Figura 5-4. Cambio en la configuración del nodo “Halifax”. Se agrega una nueva interfaz y se le brinda una dirección IP. (Fuente: elaboración propia).

Guardar los cambios en la topología (elemento B de la tabla 4-1), poner en marcha la simulación y pasar a la perspectiva de simulaciones. Esperar que cada uno de los nodos esté en estado activo y abrir una conexión Telnet para la consola de “Halifax”. Esperar a que el nodo “Halifax” esté completamente iniciado, esto es hasta que aparezcan las adyacencias BGP.

Activar el modo “Enable” (ingresar ‘ena’ en la consola). La clave de ingreso es “cisco”. Confirmar que existe la interfaz creada manualmente (“loopback10”), por medio del comando: ‘show ip int brief’. Activar el modo configuración (Configuration Mode), introduciendo el comando: ‘config t’.

Configurar la interfaz “loopback10” por medio del comando: ‘interface loopback10’, asignando una nueva dirección IP con el comando: ‘ip address 20.20.8.1 255.255.255.255’ (ver figura 5-5).

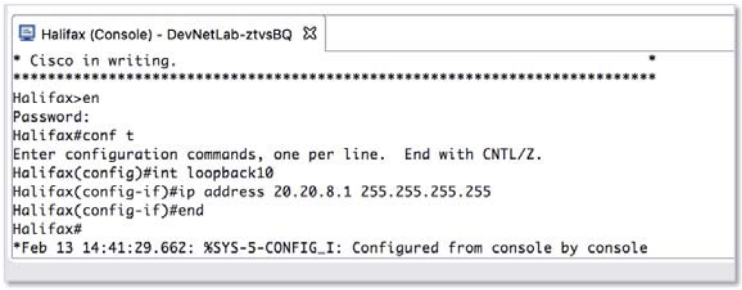


Figura 5-5. Asignación de una nueva dirección IP a la interfaz “loopback10”, mediante la consola del nodo “Halifax”. (Fuente: elaboración propia).

Restaurar el panel de Consola a su tamaño y posición original. Hacer clic en el identificador de simulaciones y seleccionar “Stop simulation”. Aparecerá un cuadro de diálogo notificando que se

detendrá la simulación. Seleccionar en este cuadro las opciones de “extraer configuración” y “cerrar todas las ventanas asociadas a la simulación” (el mensaje está en el idioma Inglés).

Confirmar la extracción de la configuración seleccionando “OK”. La extracción de la configuración y la detención de la simulación podrían tardar varios minutos.

Volver a la perspectiva de diseño y seleccionar el nodo “Halifax” para mostrar el panel de propiedades. Seleccionar la pestaña de Configuración y observar que la configuración realizada durante la simulación ha sido extraída y guardada con la topología. Tal como se ilustra en la figura 5-6.

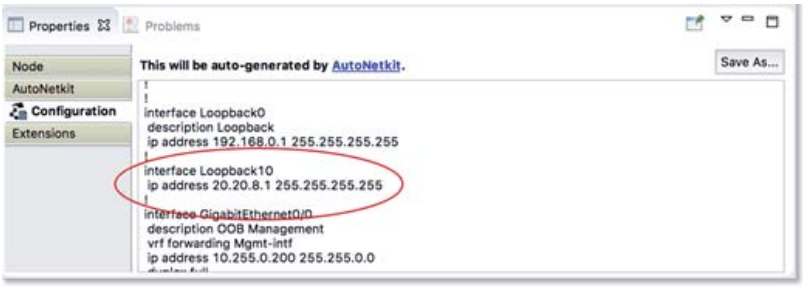


Figura 5-6. Revisión de configuración del nodo “Halifax” realizada durante la simulación (Fuente: elaboración propia).

Se concluye que es posible realizar configuraciones antes de correr la simulación, la que posteriormente será traspasada al nodo respectivo. Esta configuración puede realizarse utilizando la herramienta de configuración automática, la que también puede ser modificada manualmente como se aprecia en la figura 5-4. Además, luego de correr la simulación, se pueden realizar cambios en las configuraciones iniciales, las que serán guardadas una vez detenida la simulación, pudiendo ser visualizada en la pestaña de configuración como se muestra en la figura 5-6.

5.2 Configuración de protocolos de enrutamiento

En este bloque, se modificará la topología ilustrada en la figura 5-1, con el fin de conseguir y configurar múltiples sistemas autónomos, cada uno corriendo con diferentes IGP. eBGP se utilizará para compartir rutas entre sistemas autónomos e iBGP se utilizará en interior con sus respectivos route-reflector.

5.2.1 Experiencia N°2

Título: Configuración de protocolos de enrutamiento

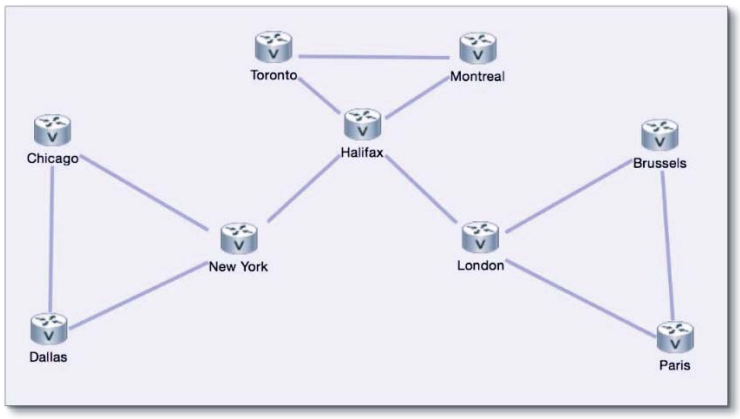


Figura 5-7. Sistemas autónomos. (Fuente: elaboración propia).

Objetivo:

Crear tres sistemas autónomos con diferentes protocolos de pasarela interior (IGP) para cada sistema. Además de utilizar BGP exterior e interior (eBGP, iBGP) para compartir rutas entre los tres sistemas.

Equipamiento:

- 9 routers IOSv.

Desarrollo:

Paso N° 1.

Crear la topología ilustrada en la figura 5-7. Seleccionar los nodos “Chicago”, “Dallas” y “New York” (Ctrl+clic), y en la pestaña AutoNetkit, en el panel de propiedades, setear ASN (Autonomous System Number) en 10, RR Cluster (Route Reflector) en 10 e IGP en “OSPF”.

Seleccionar los nodos “Toronto”, “Montreal”, y “Halifax” y setear los valores: ASN en 20, RR Cluster en 20 e IGP en “EIGRP”.

Seleccionar los nodos “Brussels”, “London”, “Paris” y asignar los valores: ASN en 30, RR Cluster en 30 e IGP en “ISIS”.

Seleccionar los nodos “Paris”, “Brussels”, “Montreal”, “Toronto”, “Chicago”, “Dallas” y asignar iBGP Role en RRC (Route Reflector Client). Además, seleccionar los nodos “New York”, “Halifax” y “London” y asignar iBGP en RR (Route Reflector).

Paso N° 2.

Reconstruir la configuración utilizando las propiedades establecidas recientemente (elemento B de la tabla 4-2). No abrir la ventana “cambios en la configuración”, se examinarán en las pestañas de configuración más adelante. Seleccionar “Yes” cuando se pregunte si se desea ver las visualizaciones AutoNetkit.

Utilizar el selector de vistas en la esquina superior izquierda de las visualizaciones AutoNetkit, para visualizar los sistemas autónomos. Seleccionando las vistas OSPF, IS-IS, EIGRP respectivamente. Además seleccionar vistas iBGP y eBGP. La figura 5-8, ilustra la visualización iBGP.

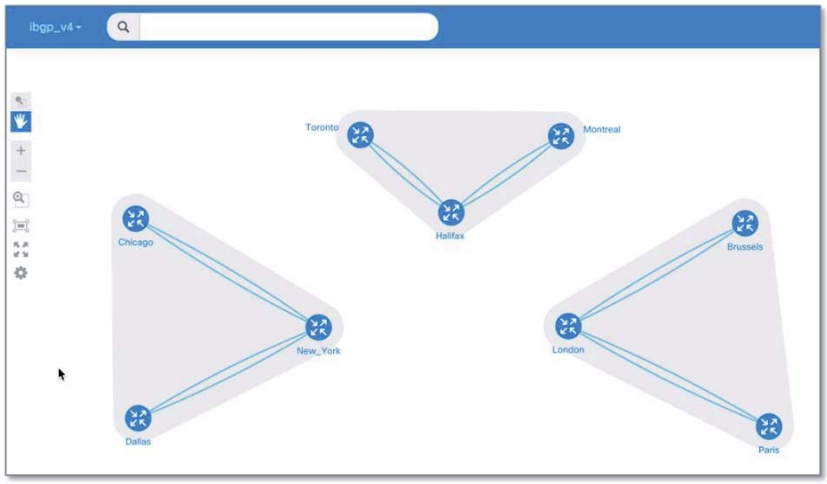


Figura 5-8. Visualización iBGP. (Fuente: elaboración propia).

En la visualización iBGP, el uso de route-reflectors es claramente visible debido a la falta de una malla completa de conexiones entre cada uno de los routers. Los route-reflector clientes (“Paris”, “Brussels”, “Montreal”, “Toronto”, “Chicago” y “Dallas”) se conectan al route-reflector respectivo (“New York”, “Halifax” y “London”), los que corresponde a Routers Peering.

En la visualización eBGP, asegurarse de que las únicas conexiones que se ven son entre sistemas autónomos. Si estas conexiones no están presentes, entonces es una clara indicación de que la numeración de sistemas autónomos (ASN) ha sido mal configurada.

Cerrar la ventana del navegador y volver a VM Maestro. Examinar los cambios en las configuraciones relacionadas con IGP y BGP, desde la pestaña configuración de los routers.

5.3 Control de nodos y estados de interfaz

La topología de la red de la figura 5-7, contiene nueve nodos IOSv. Este no es un número grande, incluso en un PC tipo laptop. Pero si se tratara de nodos IOS-XRv o la topología tuviese decenas o centenas de nodos, se estaría frente a un problema, puesto que los recursos de hardware son limitados.

VIRL posee la capacidad de excluir nodos de simulaciones, por lo que aún es posible construir topologías grandes y realizar pruebas en etapas. También es factible iniciar-detener nodos individualmente para determinar el efecto que tendría sobre la topología o una aplicación de red.

No será necesario realizar una nueva experiencia en este caso, ya que se utilizará la misma topología que en Experiencia 2. Por lo tanto, considérese como la continuación.

Objetivo:

Excluir nodos en el arranque de una simulación en la topología ilustrada por la figura 5-7. Detener y activar nodos, activar y desactivar interfaces de los nodos, además de realizar ping para corroborar lo anterior.

Paso N° 3.

Seleccionar los routers que se encuentra en el contorno de la topología: Dallas, Chicago, Toronto, Montreal, Brussels y Paris. Seleccionar la pestaña "Node" en el panel de propiedades y marcar la opción "Exclude node from simulation launch". Guardar los cambios antes de lanzar la simulación.

Correr la simulación y Observar que sólo los nodos Halifax, London, y Paris se encuentran activos, tal como se muestra en la figura 5-9.

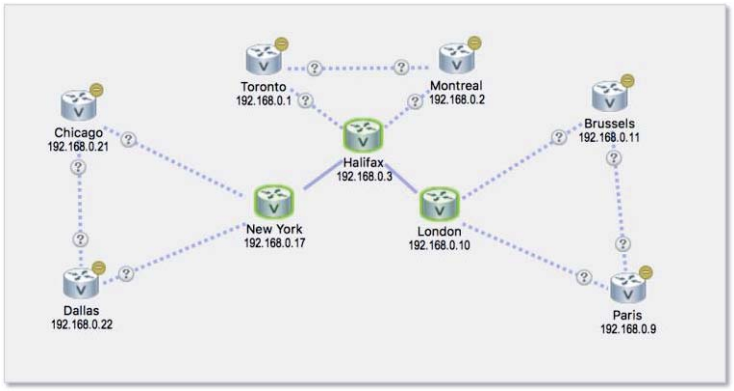


Figura 5-9. Simulación parcial. (Fuente: elaboración propia).

Abrir la consola del Router London y moverla hacia la parte de abajo del "paño activo". Cuando el Router London se haya iniciado por completo, hacer ping al nodo Halifax para confirmar la conectividad. La figura 5-10 ilustra el resultado.

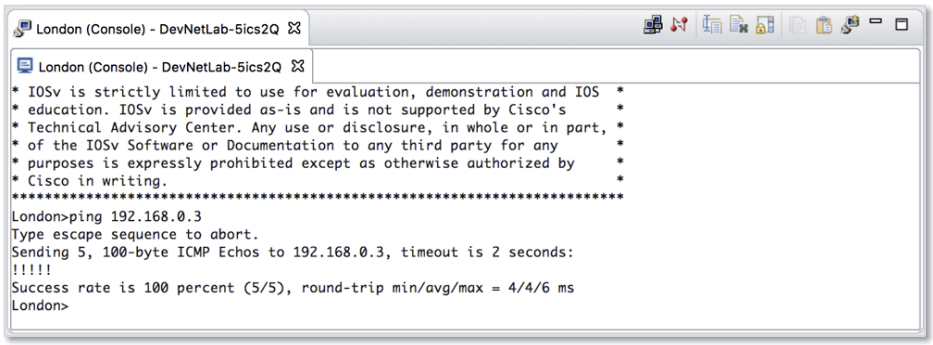


Figura 5-10. Prueba ping al Router Halifax desde el nodo London. (Fuente: elaboración propia).

Como era de esperar, Halifax está arriba. Pero si se estuviese desarrollando una nueva aplicación y se quisiera probar como respondería a un nodo fallido, VIRL permite apagar routers al igual que podría hacerse en la vida real.

Clic derecho en el nodo Halifax y seleccionar “Stop this node”. Seleccionar “OK” para confirmar que se desea detener el nodo. Esperar a que Halifax deje de brillar (ver figura 5-11).

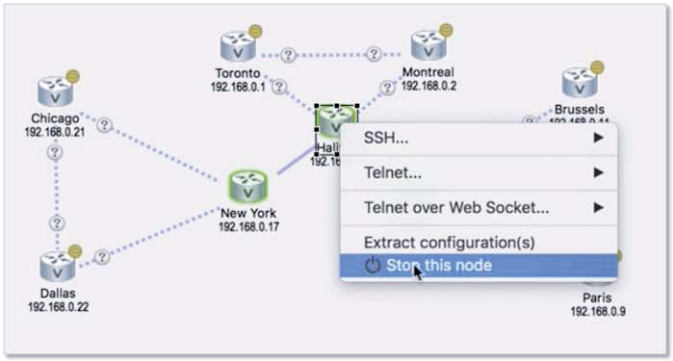


Figura 5-11. Detención de un nodo activo en una simulación. (Fuente: elaboración propia).

Comprobar que Halifax ya no está activo mediante un ping desde la consola de London. Clic derecho nuevamente en Halifax y seleccionar “Start this node”. Esperar unos minutos para que Halifax arranque y quede activo. Utilizar ping para confirmar que Halifax se encuentra activo una vez más.

Los nodos que no fueron iniciado cuando se corrió la simulación, también pueden inicializarse luego de que la simulación esté corriendo. Clic derecho en el nodo Brussels y seleccionar “Start this node”. Observar que, a medida que el nodo Brussels se activa, la conexión con London vuelve a su estado normal.

Seleccionar el enlace entre London y Brussels, y observar que aparecen conectores en ambos extremos. Hacer clic derecho en el conector que va a Brussels y seleccionar “Disable interface”, como se ilustra en la figura 5-12.

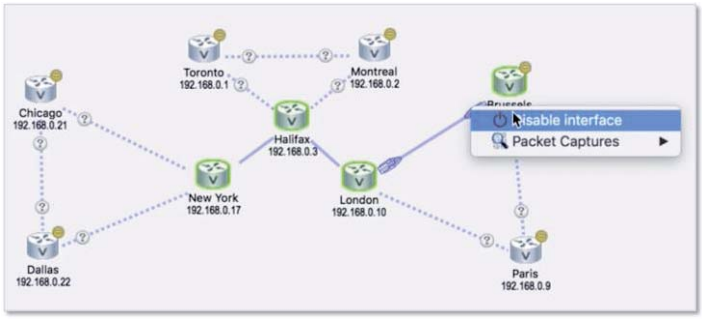


Figura 5-12. Desactivación del conector que va a Brussels. (Fuente: elaboración propia).

Observar que cuando una interfaz es desactivada, aparece una decoración de color rojo como indicador. Clic derecho en el enlace desactivado y volver a activarlo seleccionado “Enable Interface”.

Detener la simulación, cerrar las ventanas asociadas y volver a la perspectiva de Diseño.

Se concluye: realizar configuraciones como las presentadas en el paso N° 1 con equipamiento real, llevaría bastante tiempo, pero con VIRL se obtuvieron 3 sistemas autónomos con tan solo setear algunos valores en la pestaña AutoNetkit, además de la configuración de Route Reflector para compartir rutas entre los sistemas autónomos. Fue posible visualizar estas configuraciones de manera gráfica, tal como se ilustra en la figura 5-8, lo que permite una gran capacidad de análisis. Excluir nodos en el arranque de una simulación ayudará cuando se tengan topologías extensas, además detener o poner en marcha nodos permitirá realizar pruebas en escenarios de fallas.

5.4 Configuración de acceso al gestor de nodos

Ya están las condiciones para construir nuevas simulaciones que ayudarán a ilustrar las opciones de gestión de la interfaz de conexión en VIRL.

Hasta ahora se ha utilizado VM Maestro para conectarse a los puertos consola de los nodos. Conectarse con la consola es útil para realizar configuraciones, hacer pruebas y solucionar problemas.

Sin embargo, se requiere:

- Utilizar una plataforma de gestión de configuración.
- Enviar información para solución de problemas a un servidor de registro.
- Descargar y actualizar aplicaciones.
- Utilizar controladores o aplicaciones de red con una API de red.

Para cualquiera de estos requisitos, es necesario comunicarse con las interfaces de configuración de los nodos, por tanto se debe contar con conectividad IP. VIRL proporciona tres mecanismos para realizar dicha conectividad, estos son:

- Private Simulation Network.
- Private Project Network.
- Shared Flat Network.

Cada uno de estos métodos, expone la conectividad en la red “Flat”, la que se extiende desde el interior del host VIRL hacia afuera por medio de la interfaz eth1 (ver figuras 5-14, 5-15 y 5-16).

La red Flat, utiliza por defecto la subred 172.16.1.0/24 y el host VIRL posee una interfaz en la red Flat en 172.16.1.254 (ver figuras 5-14, 5-15 y 5-16).

La IP de la subred Flat, la interfaz host y la IP del host, pueden ser cambiadas cada una durante la instalación para adaptarse a las necesidades específicas del lugar, pero en este ejercicio, los valores por defecto permanecen en uso.

Los tres mecanismos que VIRL proporciona para la conectividad IP son descritos a continuación. La elección de estas opciones se realiza en la pestaña “Topology” en el panel “Properties”, tal como se ilustra en la figura 5-13.

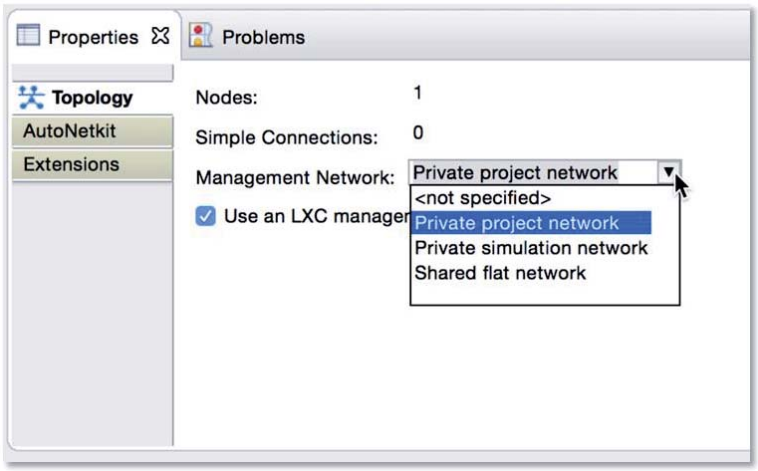


Figura 5-13. Mecanismos de conectividad que proporciona VIRL. (Fuente: elaboración propia).

5.4.1 Private Simulation, Private Project Networking y Shared Flat Networking

Cada vez que uno de los métodos privados se utiliza, es creado un LXC (Linux Containers) para cada simulación, con el fin de proporcionar un puente entre la red Flat y el gestor de red, el cual utiliza la subred 10.255.0.0/16 por defecto, en donde es colocada además cada interfaz gestora de nodo.

El acceso al gestor se consigue mediante el primer acceso de LXC utilizando protocolo SSH, y luego, utilizando Telnet o SSH desde LXC se consigue el acceso a los nodos. El LXC también puede ser configurado para enviar tráfico de los nodos en la simulación o incluso aplicaciones de red o servicios directos, al gestor de nodos en la simulación [26].

Proyectos y usuarios

Antes de entrar en más detalles respecto de los mecanismos privados de conectividad que VIRL proporciona, es importante comprender la diferencia entre proyectos y usuarios.

Un proyecto es un grupo de usuarios, y cada usuario debe estar en un proyecto. Por defecto se crea el usuario "guest" y se utiliza al conectar VM Maestro al servidor VIRL. Lo que no es evidente quizás es que este usuario "guest" ya es parte de un proyecto del mismo nombre.

Utilizando la interfaz User Workspace Management (UWM) de VIRL, es posible crear proyectos adicionales y usuarios adicionales dentro de los proyectos. Por ejemplo, se podrían crear los usuarios “pedrito” y “juan” en el proyecto “guest”, y cuando se acceda a VIRL desde VM Maestro, las credenciales utilizadas serían “guest-pedrito” y “guest-juan”, creando ambos nombres de manera evidente y lógica. Podría crearse también los proyectos “abc” o “def”, y usuarios distintos para cada uno.

Luego de comprender los conceptos de “proyectos” y “usuarios” en VIRL, se explorará la primera diferencia entre los 2 métodos privados.

Private Simulation Networking

Cuando es utilizada Private Simulation Networking, es creada una subred discreta 10.255.0.0/16 para cada simulación, y el LXC asociado (recordar que cada vez que uno de los métodos privados es utilizado se crea un LXC) tiene conectividad sólo con aquellos nodos que se ejecutan dentro de esa única simulación, tal como se ilustra en la figura 5-14.

Dentro de la línea segmentada en la figura 5-14, se aprecian los nodos visibles para el LXC respectivo de cada simulación (A y B) utilizando el mecanismo Private Simulation Networking.

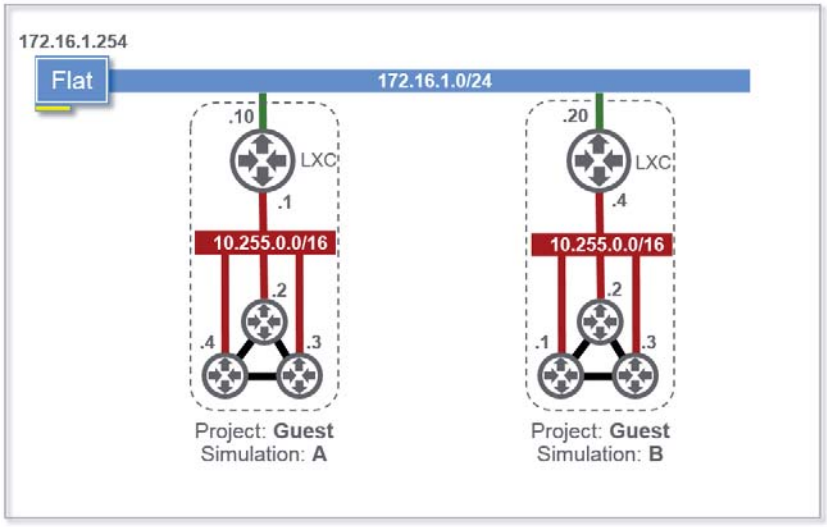


Figura 5-14. Estructura del mecanismo de conectividad Private Simulation Networking. (Fuente: http://virl-dev-innovate.cisco.com/).

En cualquier otra simulación el LXC no puede ver a los nodos y por tanto no puede acceder a ellos, incluso los que se ejecuta como parte del mismo proyecto, tal como se ilustra la figura 5-14.

Private Project Networking

Cuando se utiliza Private Project Networking, se crea una subred única discreta 10.255.0.0/16 y los LXCs que acompañan a cada simulación tienen conectividad a todos los nodos que se ejecutan dentro de dicho proyecto, independiente del usuario que posea la simulación.

La línea segmentada en la imagen 5-15 representa la visibilidad que posee el LXC sobre los respectivos nodos. Notar que en este caso hay 3 simulaciones y 2 proyectos distintos.

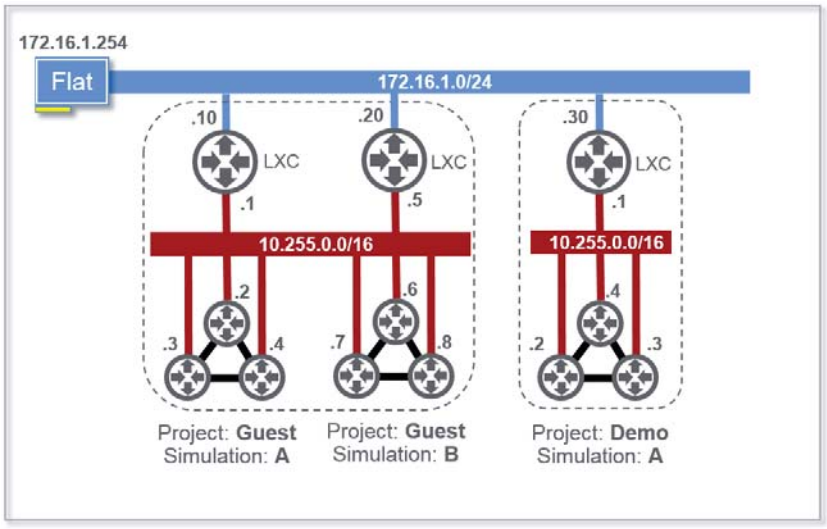


Figura 5-15. Estructura del mecanismo de conectividad Private Project Networking. (Fuente: http://virl-dev-innovate.cisco.com/).

Como se ve en la figura 5-15, en cualquier otro proyecto, los LXCs no pueden ver a los nodos y por lo tanto no pueden acceder a ellos (el LXC del proyecto “Demo” y la simulación “A” no podrá ver a los nodos del proyecto “Guest”). Tampoco se podrá acceder a los nodos que forman parte de simulaciones que utilizan Private Simulation Networking, incluso los nodos que están en el mismo proyecto.

Shared Flat Networking

El mecanismo Shared Flat Networking difiere principalmente de los 2 mecanismos Privados en:

- Las interfaces de administración de los nodos en una simulación son colocados directamente sobre la red Flat (172.16.1.0/24).
- No es creada la subred 10.255.0.0/16 y tampoco son creados o utilizados los LXCs para el acceso a la administración.
- Los nodos tienen visibilidad a todos los otros nodos en la simulación utilizando Shared Flat Networking independientemente del proyecto o el usuario.

- Los nodos tienen acceso a todos los LXCs asociados con otras simulaciones independientemente del proyecto o usuario.
- Los nodos tienen conectividad directa a todos los dispositivos de la red Flat incluyendo host VIRL y dispositivos accesibles desde “eth1”.

La figura 5-16 ilustra lo descrito anteriormente.

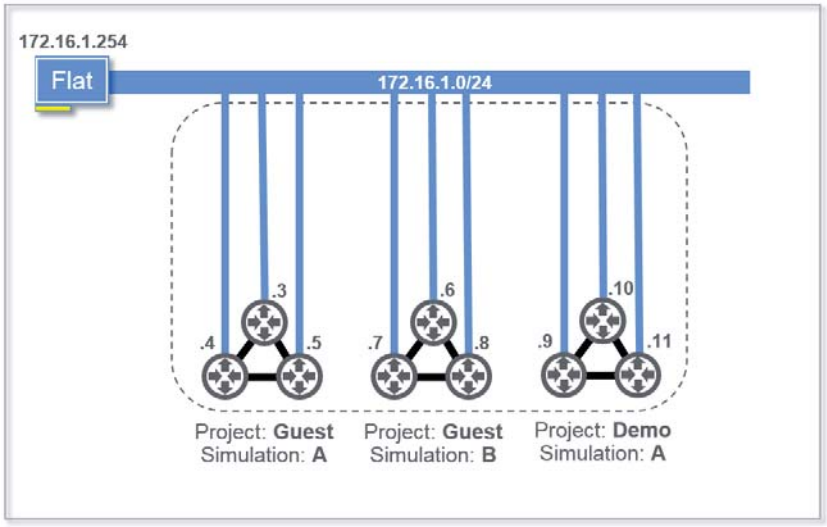


Figura 5-16. Estructura del mecanismo de conectividad Shared Flat Networking. (Fuente: http://virl-dev-innovate.cisco.com/).

En la figura 5-16, se aprecian 3 simulaciones y 2 proyectos. Donde los nodos en cada simulación, independiente del proyecto y la simulación, pueden verse todos entre sí.

5.4.2 Experiencia N°3

Ahora que se conoce las diferencias entre los tres mecanismos de acceso a la gestión de nodos, se realizará una actividad con VIRL para ilustrar lo visto anteriormente.

En este ejercicio se crearán cuatro nuevas topologías que contienen un único Router IOSv, donde serán utilizados todos y cada uno de los mecanismos de gestión de red descritos anteriormente.

A continuación se accederá al host VIRL directamente con el fin de obtener acceso a la red Flat y examinar la accesibilidad que existe entre los nodos en las cuatro simulaciones.

Título: Mecanismos de conectividad.

Objetivo:

Conectarse a los LXCs de métodos privados de simulación y probar la visibilidad que poseen estos según 5.4.1. Realizar pruebas de visibilidad en el método de simulación Shared Flat Networking. Por medio de la terminal en el host VIRL.

Nota: en esta experiencia se deberá anotar algunas direcciones IP, por lo que debe tenerse a mano un blog de notas o lápiz y papel.

Equipamiento:

- 1 Router IOSv, método de simulación Private Simulation Network.
- 1 Router IOSv, método de simulación Private Project Network.
- 1 Router IOSv, método de simulación Private Project Network.
- 1 Router IOSv, método de simulación Shared Flat Network.

Desarrollo:

Paso N° 1.

Crear y poner en marcha cuatro nuevas topologías, cada una con un único Router IOSv y realizar la siguiente configuración:

- 1° topología: nombre “Lab.09.private.virl” utilizando Private Simulation Network.
- 2° topología: nombre “Lab.09.project.1.virl” utilizando Private Project Network.
- 3° topología: nombre “Lab.09.project.2.virl” utilizando Private Project Network.
- 4° topología: nombre “Lab.09.shared.virl” utilizando Shared Flat Network.

Generar configuraciones para cada topología y nombrar los Routers de manera que coincida con la topología a la que pertenece para que sea más fácil reconocerlos más adelante.

Paso N° 2.

Utilizando el panel de simulación, identificar y anotar la dirección IP asociada con el nodo IOSv en la simulación “private”:

a) Clic derecho en el nodo
b) Seleccionar “Telnet…”
c) Buscar la dirección asociada al puerto “Management”.

En este ejemplo la dirección IP asociada con el nodo IOSv en la simulación “private” es 10.255.0.3, tal como se ve en la figura 5-17.

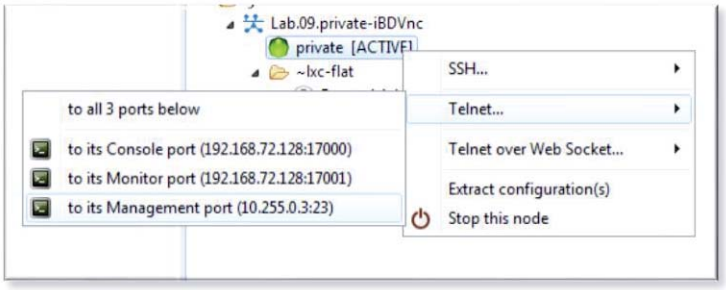


Figura 5-17. Dirección IP asociada al nodo private en simulación “private”. (Fuente: elaboración propia).

Nota: no es posible acceder al puerto “to its Management port” con un simple clic, puesto que de esta manera no funcionará. La dirección IP que se muestra corresponde a una subred que existe en el host VIRL y no se podrá acceder sin antes haber realizado una configuración especial, configuración que no se aborda en este capítulo.

Identificar (en el panel de simulación) y anotar la dirección IP asociada con el LXC creado por la simulación “private”. En este ejemplo la dirección IP del LXC es 172.16.1.89, tal como se ilustra en la figura 5-18.

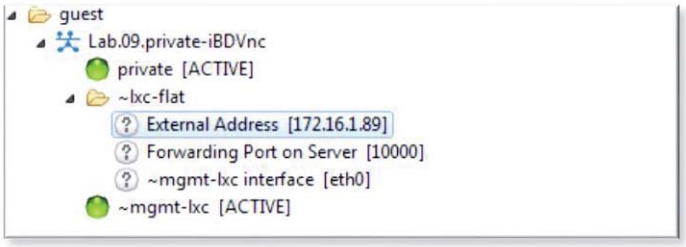


Figura 5-18. Dirección IP del LXC asociado a la simulación “private”. (Fuente: Laboratorio Digitales A, EIE PUCV).

De la misma manera anotar las direcciones IP asociadas a los nodos IOSv y LXCs en “Project.1” y “Project.2”.

Registrar (anotar) la dirección IP “Management” del nodo IOSv en la simulación “shared”. En este ejemplo la dirección IP “Management” del nodo IOSv en la simulación es 172.16.1.92. (ver figura 5-19).

Figura 5-19. Dirección IP asociada al nodo shared en simulación “shared”. (Fuente: elaboración propia).

LXCs en simulaciones "shared".

Observar que la dirección IP de administración (Management en la figura 5-19) para el nodo IOSv en la simulación "shared" está en la subred Flat al igual que los LXCs asociados a las simulaciones “privadas”. Debido a esto no se necesita LXC para obtener acceso de administración. Aun así LXCs son creados para simulaciones "shared", pero se utilizan para otros fines tales como la extracción de configuración y pueden ignorarse aquí.

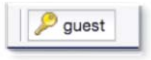
Luego de haber registrado las direcciones IP que se necesitan en cada una de las 4 simulaciones, se requiere acceder a la subred Flat para poder conectar a los distintos LXCs y nodos IOSv. Hay

varias maneras de conseguir acceso a la subred Flat, incluso hay una que se utiliza todo el tiempo y por todo usuario de VIRL, es la que permite iniciar sesión en el host VIRL. Esta interfaz se usa de manera permanente y directa dentro de la subred Flat.

Paso N° 3.

Encontrar la dirección IP del host VIRL. Esto se logra fácilmente a través de "Credentials Tool", en la parte inferior derecha de la barra de estados de VM Maestro.

Clic en el botón:



A continuación, buscar la dirección IP (cuadro amarillo en la figura 5-20). En este ejemplo la dirección IP del host VIRL es 192.168.72.128.

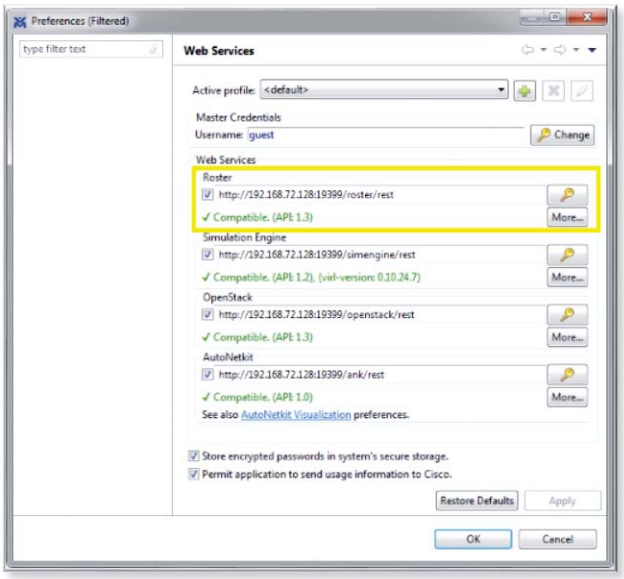


Figura 5-20. Credentials Tool. (Fuente: elaboración propia).

Se debe utilizar una herramienta SSH apropiada para la plataforma de conexión con el host VIRL, en este caso, se está utilizando el sistema operativo Windows como anfitrión, por lo que es necesario descargar del sitio web http://www.putty.org/ la herramienta PuTTY. PuTTY es un cliente SSH, Telnet, rlogin, y TCP raw con licencia libre, por lo que permitirá la conexión con el host VIRL mediante cliente SSH. El nombre de usuario y la contraseña para acceder al host VIRL es "virl" y "VIRL" respectivamente.

En la figura 5-21, se aprecia la dirección IP del host VIRL al que se accederá. Asegurarse de que la opción SSH se encuentre seleccionada. Luego hacer clic en "Open". Se abrirá una ventana donde solicita la clave de acceso al host VIRL (password: VIRL). La figura 5-22 ilustra esto con claridad.

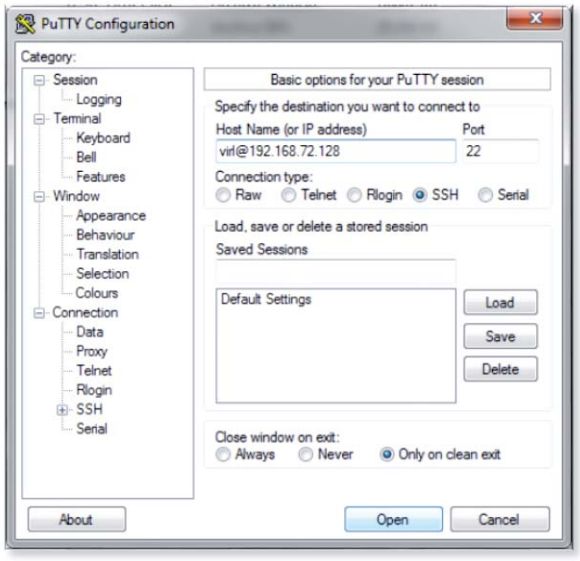


Figura 5-21. PuTTY como cliente SSH. (Fuente: elaboración propia).

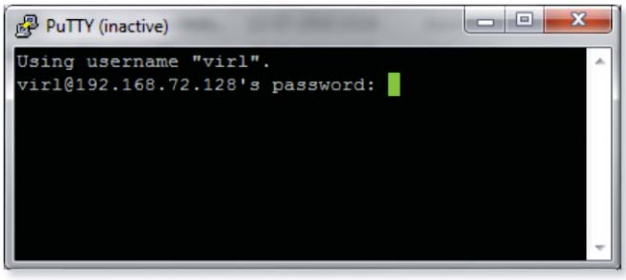


Figura 5-22. Terminal PuTTY. Conexión con el host VIRL. (Fuente: elaboración propia).

Una vez conectado al host VIRL, utilizar SSH para conectarse al LXC asociado a la simulación “private” utilizando el nombre de usuario “guest” y contraseña “guest”. Durante el uso de SSH para conectarse al host VIRL o los LXCs, podría solicitarse que se acepte la clave RSA asociada con el destino. Es seguro y necesario aceptar estas claves mediante la introducción de "yes".

Una vez conectado al LXC, utilizar Telnet para conectarse a la interfaz de gestión del nodo IOSv en la simulación “private”. La dirección IP asociada con el nodo IOSv es 10.255.0.3, obtenida anteriormente. El nombre de usuario y contraseña es “cisco”. Ver figura 5-23.

Figura 5-23. Telnet para conectarse a la interfaz de gestión del nodo IOSv en la simulación “private”. (Fuente: elaboración propia).

Salir del nodo IOSv cerrando la conexión Telnet con el comando “exit”. Se retornará al LXC asociado a la simulación “private”. Luego, intentar entrar a las interfaces de gestión de los nodos IOSv en las simulaciones “Project.1” y Project.2”. Podrá observarse que no es posible acceder a estos, tal y como se esperaba.

Salir del LXC asociado a la simulación “private” con el comando “exit” y utilizando protocolo SSH, conectarse a uno o ambos LXCs asociados con las simulaciones “project.1” y “project.2”. Mediante telnet, confirmar que existe conectividad desde cualquiera de los LXCs asociados a las simulaciones “project”, a los nodos IOSv en cualquiera de ambas simulaciones, y confirmar que no existe conectividad desde cualquiera de los LXCs asociados a las simulaciones “project”, al nodo IOSv en la simulación “private”. Salir del LXC (o LXCs) asociados a las simulaciones “project”.

Desde el host VIRL (máquina virtual en VMware Workstation), abrir la terminal (xterm) y utilizar Telnet para conectarse directamente al nodo IOSv de la simulación “shared”. Esto se aprecia en la figura 5-24.

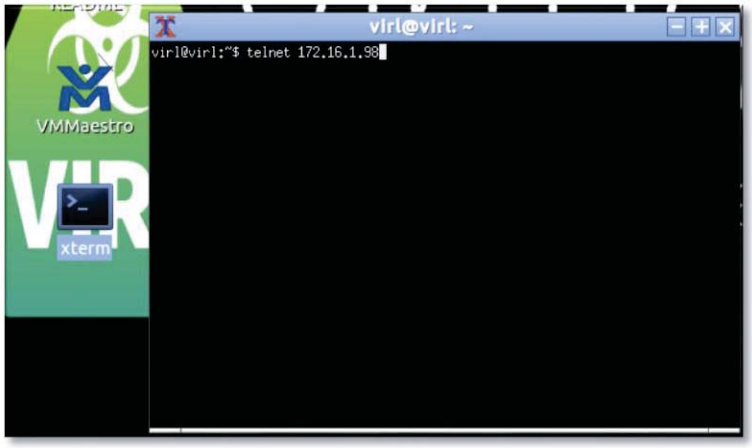


Figura 5-24. Telnet desde la terminal de la máquina virtual VIRL. (Fuente: elaboración propia).

Observar que el acceso al nodo IOSv en la simulación “shared” se puede hacer directamente por medio de la red Flat sin la necesidad de utilizar un LXC. Realizar un Ping desde el nodo IOSv de la simulación “shared”, para confirmar conectividad a otros dispositivos en la red Flat, tal como los LXCs asociados a las simulaciones “private” y “project”. Salir del Router IOSv asociado con la simulación “shared”, salir del host VIRL, Volver a VM Maestro y detener todas las simulaciones.

Se concluye que, mediante las pruebas realizadas, es posible confirmar la exposición que poseen los tres mecanismos de simulación representados por las figuras 5-14 para el caso de “Private Simulation Networking”, 5-15 para “Private Project Networking” y la figura 5-16 para el método “Shared Flat Networking”. En resumen, para “Private Project Networking”, se tiene acceso a todos los nodos de un mismo proyecto independiente de la simulación, no así para proyectos distintos; para “Private Simulation Networking”, sólo es posible acceder a los nodos dentro de la misma simulación y no a nodos de otras simulaciones, aunque pertenezcan al mismo proyecto; y por

último, el método “Shared Flat Networking”, permite acceder a cualquier nodo, de cualquier proyecto y cualquier simulación.

5.5 Configuración de conmutadores de capa 2

Hasta el momento se han construido sólo topologías con Routers conectados mediante enlaces punto a punto, tal como las implementaciones que deberían encontrarse en áreas extensas. Sin embargo, la mayoría de las redes incluyen cantidades significativas de conmutadores de capa 2 (Switch) en conjunto con routers, conectados en redes multipunto comunes [26].

VIRL incluye 2 mecanismos de configuración de conmutadores de capa 2, estos son:

- Un subtipo de conmutador no administrado que se comporta como un conmutador Ethernet con un único dominio de difusión.
- Un subtipo de conmutador IOSvL2 administrado, que se configura y se comporta como una plataforma IOS de capa 2.

5.5.1 Experiencia N°4

Título: Configuración de dispositivos de capa 2.

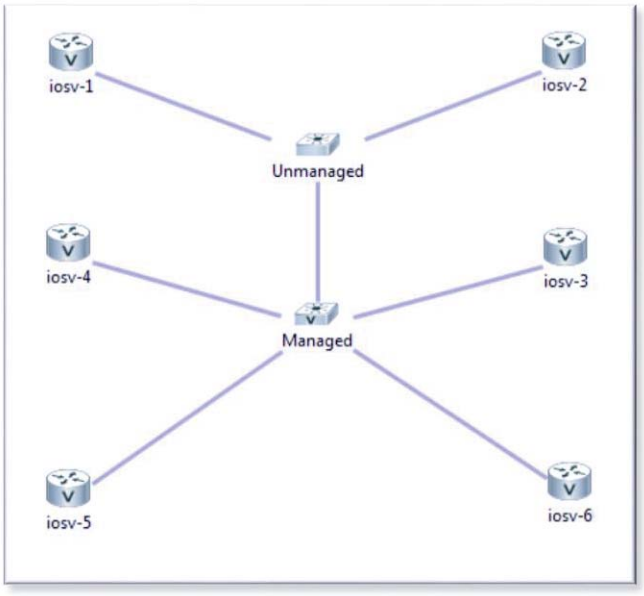


Figura 5-25. Conexiones de la topología. (Fuente: elaboración propia).

Objetivo:

Utilizar ambos tipos de conmutadores para crear una red OSPF [27] sencilla, con un único dominio de difusión de manera tal que cada Router forme adyacencias con todos los demás. Además, se creará otra red utilizando VLANs (redes de área local virtuales) sobre conmutadores

IOSvL2 administrados, con la finalidad de crear 2 dominios de difusión y observar el efecto que esto tiene sobre la formación de adyacencias.

Equipamiento:

- 6 nodos IOSv
- 1 Switch no administrado
- 1 Switch IOSvL2

Desarrollo:

Paso N°1.

Crear una nueva topología, nombrarla como se desee. Asegurarse de que el nombre termine en "nombre**.virl**" y realizar la topología de la figura 5-25. Generar las configuraciones y correr la simulación.

Paso N°2.

Cuando la simulación muestre que todos los nodos están activos, observar que el conmutador "Managed" se incluye en la lista de nodos en el panel de simulación y el conmutador "Unmanaged" no es incluido en esta lista. Abrir una conexión con el puerto consola del nodo "iosv-1". Esperar a que el Router se haya iniciado correctamente.

Introducir el comando 'show ip ospf ne' y observar que se han formado 5 adyacencias OSPF, una por cada uno de los otros Router IOSv en la simulación. Abrir una conexión con el puerto consola del nodo "iosv-6" y observar que posee 5 adyacencias al igual que el nodo "iosv-1". Ver figuras 5-26 y 5-27.

```
iosv-1#show ip ospf ne

Neighbor ID     Pri   State           Dead Time   Address         Interface
192.168.0.2       1   FULL/DROTHER    00:00:37    10.0.0.2        GigabitEthernet0/1
192.168.0.3       1   FULL/DROTHER    00:00:38    10.0.0.3        GigabitEthernet0/1
192.168.0.4       1   FULL/BDR        00:00:38    10.0.0.4        GigabitEthernet0/1
192.168.0.5       1   FULL/DROTHER    00:00:38    10.0.0.5        GigabitEthernet0/1
192.168.0.6       1   FULL/DROTHER    00:00:39    10.0.0.6        GigabitEthernet0/1
iosv-1#
```

Figura 5-26. Adyacencias OSPF de nodo iosv-1. (Fuente: elaboración propia).

```
iosv-6#show ip ospf ne

Neighbor ID     Pri   State           Dead Time   Address         Interface
192.168.0.1       1   FULL/DR         00:00:35    10.0.0.1        GigabitEthernet0/1
192.168.0.2       1   2WAY/DROTHER    00:00:38    10.0.0.2        GigabitEthernet0/1
192.168.0.3       1   2WAY/DROTHER    00:00:38    10.0.0.3        GigabitEthernet0/1
192.168.0.4       1   FULL/BDR        00:00:36    10.0.0.4        GigabitEthernet0/1
192.168.0.5       1   2WAY/DROTHER    00:00:34    10.0.0.5        GigabitEthernet0/1
iosv-6#
```

Figura 5-27. Adyacencias OSPF del nodo iosv-6. (Fuente: elaboración propia).

Detener la simulación, cerrar todas las consolas y volver a la perspectiva de diseño.

Paso N°3.

Seleccionar la conexión que hay entre el conmutador “Managed” y el nodo “iosv-6” y en el editor de nodos (“Node Editor”), seleccionar la interfaz GigabitEthernet en el nodo “iosv-6”. Ver figura 5-28.

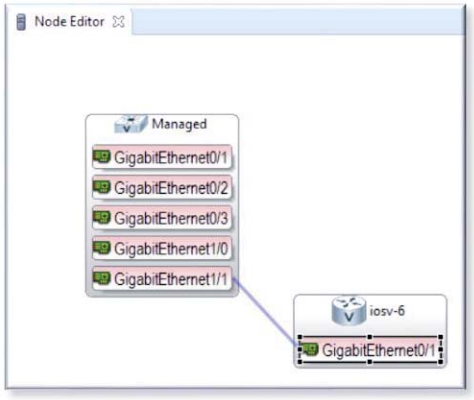


Figura 5-28. Selección de la interfaz GigabitEthernet 0/1 del nodo iosv-6 en el editor de nodos. (Fuente: elaboración propia).

Setear VLAN en “10”, esto en la pestaña “AutoNetkit” en el panel de propiedades. Repetir lo mismo para el Router “iosv-5”.

Nota: La configuración de VLANs en VIRL se define en los nodos en lugar de los conmutadores, esto para una mayor comprensión y facilidad de configuración. AutoNetkit maneja los detalles de configuración.

Generar las configuraciones y correr la simulación. Cuando la simulación muestre que todos los nodos están activados, conectarse a la consola de “iosv-1” e introducir el comando para visualizar las adyacencias que se formaron y observe que se han creado 3 adyacencias OSPF. Los nodos “iosv-5” e “iosv-6” ya no son adyacentes, puesto que han sido colocados en la VLAN 10.

Abrir la consola del nodo “iosv-6”, introducir el comando para visualizar las adyacencias OSPF y observar que se ha formado sólo una adyacencia con el nodo “iosv-5” puesto que ambos están en la VLAN 10. Abrir la consola del conmutador “Managed”, introducir el comando ‘show vlan’ y observar la configuración de VLAN que se utiliza para aislar los Routers “iosv-5” e “iosv-6” del resto de la red.

Detener la simulación, cerrar todas las consolas y retornar a la perspectiva de diseño.

Claramente, realizar estas configuraciones con equipamiento real tomaría bastante tiempo, además de poseer el manejo necesario en comandos para realizar dichas configuraciones. Sin embargo, con VIRL es posible conseguir VLANs tan solo con la introducción de parámetros, ya que AutoNetkit se encarga de realizar estas configuraciones. Esto presenta una clara ventaja,

puesto que para fine educativos y empresariales, el tiempo se orienta a la compresión más que en la programación de los dispositivos, ya que esta tarea es realizada por técnicos, quienes no poseen la tarea de realizar análisis de redes necesariamente.

5.6 Utilización de nodos Específicos

Hasta ahora se han utilizado sólo nodos IOSv e IOSvL2. Este tipo de nodos poseen bajos requerimientos de memoria y procesador. Por lo tanto la inicialización para estos tipos de nodos es rápida en comparación a otros subtipos de nodos.

Recordar que se está utilizando un PC con procesador Intel Core i5-4440 (cuatro núcleos) y se dispone de un total de 16 [GB] de memoria RAM. A la máquina virtual VIRL, fueron asignados 4 núcleos y 9.8 [GB] de memoria RAM. En base a estos recursos de hardware, es posible realizar simulaciones simultáneas de uno o dos subtipos de nodos IOS XRv, CSR1000v, NX-OSv, o ASAv, en conjunto con algunos nodos IOSv o Servidores. Si se pretende utilizar más de los nodos mencionados anteriormente, el lanzamiento de la simulación caerá en una falla debido a la falta de memoria.

La tabla 5-1, ilustra los requerimientos de hardware para los distintos tipos de nodos disponibles en el software [26].

Tabla 5-1. Requerimientos de Hardware para subtipos de Nodos VIRL.

Subtipo de Nodo.	CPU v.	Memoria RAM.
IOSv	1 CPU virtual	512 [MB]
IOSvL2	1 CPU virtual	768 [MB]
IOS-XRV	1 CPU virtual	3 [GB]
CSR1000v	1 CPU virtual	3 [GB]
NX-OSV	1 CPU virtual	3 [GB]
ASAV	1 CPU virtual	2 [GB]
Servidor (Ubuntu 14.04)	1-8 CPU virtual	512 [MB] - 8 [GB]
Linux Container (LXC)	n / A	n / A

5.6.1 Experiencia N°5

Título: Utilización de nodos específicos.

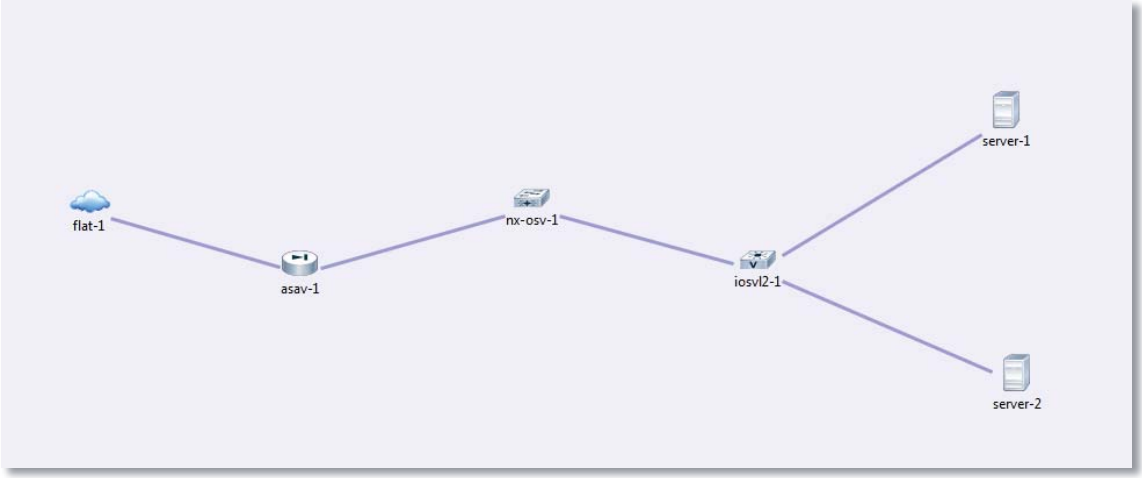


Figura 5-29. Pequeño segmento de una topología de un centro de datos con arquitectura Top-of-Rack. (Fuente: elaboración propia).

Objetivo:

Crea un segmento pequeño de una topología de un centro de datos con arquitectura Top-of-Rack, tal como el que se muestra en la figura 5-29. Utilizar la subred Flat para proporcionar conectividad exterior para data-plane a interfaz ASAv. Una vez que la red esté arriba y corriendo, configurar el nodo ASAv de tal manera de proporcionar NAT a dispositivos en la simulación. Configurar un Router estático end-to-end para permitir conexión SSH entre los servidores y el "mundo exterior".

Nota:

Utilizar la subred Flat, como la que muestra en la figura 5-29 (la nuve), para proporcionar conectividad, implica que deberá ser utilizado uno de los métodos privados de simulación, puesto que el mecanismo de simulación "Shared flat network" no permite la conectividad de data-plane y gestionar los nodos al mismo tiempo.

Se refiere a conectividad exterior dentro de la misma estructura de VIRL. Conectarse con redes exteriores reales será visto en el próximo capítulo, el principio de funcionamiento es el mismo, sólo que deben considerarse otras cosas.

Al igual que en los ejercicios anteriores, se utilizará AutoNetkit para realizar las configuraciones básicas, pero se necesitará personalizar esta configuración para llevar a cabo los objetivos.

Equipamiento:

- 2 Servidor Ubuntu
- 1 Switch IOSvL2
- 1 Switch NX-OSv
- 1 firewall ASAv
- 1 Interfaz Flat1

Desarrollo:

Paso N° 1.

Crear una nueva topología con el nombre que se desee y seleccionar el mecanismo de simulación "Private simulation network". Crear la topología que se ilustra en la figura 5-29.

Seleccionar el enlace que une a ASAv y NX-OSv y en el panel editor de nodos, seleccionar la interfaz GigabitEthernet que se conecta al nodo NX-OSv. Ver figura 5-30.

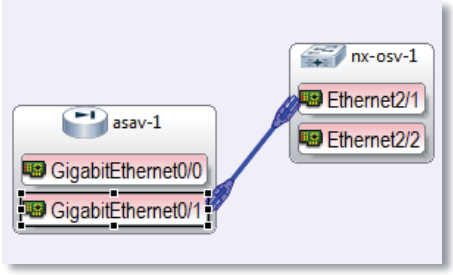


Figura 5-30. Interfaz GigabitEthernet 0/1 del nodo ASAv. (Fuente: elaboración propia).

En la pestaña AutoNetkit, setear el valor de nivel de seguridad en 100. El rango va desde 0 a 100, donde 100 es el máximo (ver figura 5-31). Para más información sobre nivel de seguridad visitar la referencia [28].

Figura 5-31. Selección de nivel de seguridad para interfaz GigabitEthernet 0/1 del nodo ASAv. (Fuente: elaboración propia).

Generar configuraciones, seleccionar el nodo NX-OSv y examinar la configuración generada. Localizar y registrar la dirección IP de la interfaz que se conecta a ASAv. En este caso la dirección IP es 10.0.128.2, tal como se ilustra en la figura 5-32.

Figura 5-32. Dirección IP asignada a la interfaz Ethernet 2/1 del nodo NX-OSv. (Fuente: elaboración propia).

Localizar y registrar la dirección IP de la interfaz que se conecta al conmutador IOSvL2. En este caso, la dirección IP es 10.0.0.1.

Seleccionar el nodo ASAv y examinar la configuración generada. Localizar y registrar la dirección IP de la interfaz "interior" que está conectada al nodo NX-OSv. En este caso, la dirección IP es 10.0.128.1. Lanzar la simulación y esperar a que todos los nodos se encuentren en estado "activo".

Paso N° 2.

Abrir la consola del nodo ASAv y esperar a que se inicialice por completo. Entrar al modo "enable" introduciendo 'en'. La Password es "cisco". Mostrar y examinar las direcciones IP que se establecieron por medio de DHCP para interfaz "outside" y vía configuración para interfaz "inside".

Entrar en modo configuración introduciendo el comando 'config t'. Configurar NAT para crear traducciones dinámicas para el tráfico que se origina en la interfaz "inside", con destino a la interfaz "outside". El comando es 'nat (inside,outside) after-auto source dynamic any interface'.

Configurar una ruta IP estática en la interfaz "inside" para la red 10.0.0.0/8 (direcciones de infraestructura VIRL) apuntando hacia NX-OSv. Recordar que esta dirección IP fue registrada anteriormente: 10.0.128.2. El comando para realizar la configuración es 'route inside 10.0.0.0 255.0.0.0 10.0.128.2'

Abrir y maximizar la consola del nodo NX-OSv. Usuario y contraseña es "cisco". Entrar al modo configuración por medio del comando "config t". Configurar una ruta IP por defecto apuntado hacia el nodo ASAv. La dirección IP fue registrada es: 10.0.128.1. El comando es 'ip route 0.0.0.0 0.0.0.0 10.0.128.1'.

Abrir una consola para cualquiera de los dos servidores. Usuario y contraseña es "cisco". Configurar una ruta IP por defecto apuntando hacia la interfaz del nodo NX-OSv que conecta con el nodo IOSvL2. Según lo registrado la dirección IP es: 10.0.0.1. El comando es 'sudo route add default gw 10.0.0.1'

Paso N° 3.

Utilizando SSH, conectarse a la interfaz "Flat" en el host VIRL para confirmar que la configuración funciona de manera apropiada. Recordar que la dirección IP asociada a la interfaz del host VIRL

es 172.16.1.254. Para completar el acceso, aceptar “RSA key” y utilizar la contraseña “VIRL”. El comando es ‘ssh virl@172.16.1.254’. Una vez conectado, volver a la consola ASAv.

Mostrar la tabla de traducción NAT para confirmar que fue creada una traducción dinámica para la conexión SSH que se realizó entre el servidor-2 y el host VIRL. Utilizar el comando ‘show xlate’.

La figura 5-33 ilustra la respuesta al comando ‘show xlate’.

Figura 5-33. Tabla de traducción NAT para conexión SSH que se realizó entre el servidor-2 y el host VIRL. (Fuente: elaboración propia).

Detener la simulación y marcar los campos para extraer la configuración y cerrar las consolas que fueron abiertas.

Se concluye que resulta muy simple realizar este tipo de topologías con VIRL, además de poder apreciar las características propias de cada nodo específico y su funcionamiento, como las traducciones NAT. Las facultades que brinda la utilización de la nuve “flat-1”, permite realizar este tipo de pruebas donde es posible conectarse con el “mundo exterior”.

Resumen

Se logró que el usuario se familiarice con las variadas posibilidades de realizar configuraciones en nodos IOSv (routers). Se crearon tres sistemas autónomos con diferentes protocolos de pasarela interior (IGP) para cada sistema. Además de utilizar BGP exterior e interior (eBGP, iBGP) para compartir rutas entre los tres sistemas.

Fueron excluidos nodos en el arranque de una simulación, además de detener y activar nodos, activar y desactivar interfaces de los nodos, además de realizar ping para realizar pruebas.

Fueron revisados y descritos los tres mecanismos de simulación que VIRL posee.

Se realizaron conexiones a los LXCs creados en los métodos privados de simulación, además de la comprobación de visibilidad que poseen estos según sus características vistas. También se realizaron pruebas de visibilidad para el método de simulación Shared Flat Networking, por medio de la terminal en el host VIRL.

Se realizaron configuraciones para dispositivos de capa dos. Además se realizaron configuraciones para crear VLANs (redes de área local virtuales) sobre conmutadores IOSvL2.

Por último, se creó un segmento pequeño de una topología de un centro de datos con arquitectura Top-of-Rack. Donde se utilizó la subred Flat para proporcionar conectividad exterior para data-plane a interfaz de nodo ASAv. Realizando configuraciones adicionales para proporcionar NAT a dispositivos en la simulación.

6 Características avanzadas

A continuación, se presentan temas relacionados a características avanzadas del software. Específicamente, se abordarán tres tópicos principales, estos son: APIs VIRL y automatización de funciones, herramientas avanzadas para utilizar en simulaciones como captura de paquetes, introducción de latencia, jitter y pérdida de paquetes, entre otros. Por último se expone sobre la estructura del software para concluir con conectividad exterior.

6.1 APIs VIRL y automatización de funciones

VIRL depende en gran medida de la utilización de APIs (*Application Programming Interfaces*, Interfaz de Programación de Aplicaciones) que soportan las comunicaciones entre varios subsistemas y aplicaciones de VIRL.

Las APIs siempre están presentes y son utilizadas por cada elemento principal de VIRL [26], incluyendo:

- El sistema OpenStack IaaS, incluye Nova, Neutron, Glance, Keystone y otros servicios OpenStack (múltiples puertos TCP).
- El Motor de Simulación, quien controla la programación, la gestión y las redes de los nodos en las simulaciones (puerto TCP 19399).
- Roster, quién expone información sobre estados de la simulación (puerto TCP 19399).
- El motor de configuración AutoNetkit, el cual realiza las configuraciones iniciales para las topologías (puerto TCP 19399).
- El Administrador de Área de Trabajo de Usuario (*UWM, User Workspace Manager*), el cual proporciona a VIRL un panel de gestión del sistema (puerto TCP 19400).
- El Servicio de Visualización AutoNetkit, quien proporciona información estática y en tiempo real sobre el estado de varios protocolos de red (puerto TCP 19401).

Las APIs proporcionadas tanto por OpenStack y VIRL, son APIs que pertenecen a la clase "RESTful" (*Representational State Trasfer*). Este tipo de API consiste en una arquitectura de software-comunicación (ver figura 6-1) que define la manera de cómo operará. Fue desarrollada para mejorar la escalabilidad, el rendimiento y la portabilidad de las aplicaciones basadas en la Web [29]. La figura 6-1 ilustra un esquema conceptual sobre las RESTful APIs en el Software VIRL.

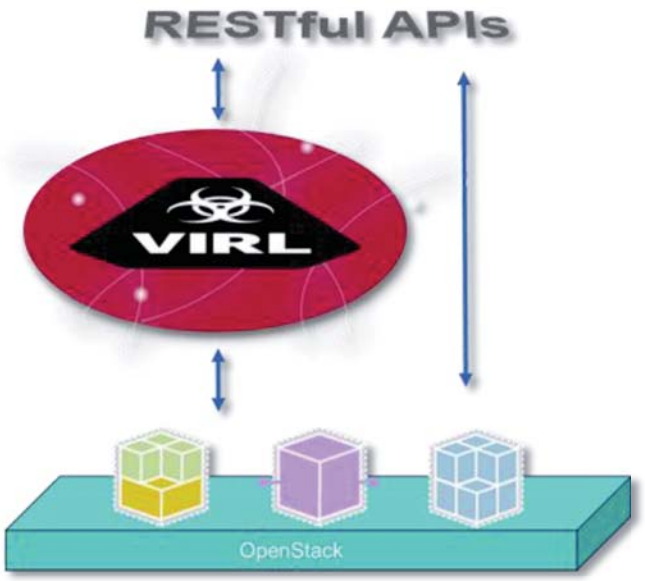


Figura 6-1. Arquitectura de RESTful APIs en VIRL (fuente: http://virl-dev-innovate.cisco.com/).

Es posible encontrar una gran cantidad de información sobre interfaces RESTful API, de hecho se han escrito libros enteros al respecto. Sin embargo, para el propósito, basta con revisar algunas de las principales características:

- Se basan en cliente-servidor. Los clientes envían peticiones (a veces con datos) a los servidores, los cuales responden a las peticiones con datos y/o información de estados.
- Utilizan HTTP como protocolo de transporte. Ya sea HTTP 1.1 o más recientemente HTTP 2.0 (SPDY).
- Utilizan URIs por HTTP (típicamente URLs), para definir e identificar objetos.
- Utilizan Verbos (Verbs) o "métodos" por HTTP, para describir acciones. Algunos métodos más comunes son:
    - GET para recuperar los datos o para activar un evento o acción.
    - PUT para reemplazar un conjunto de datos con un conjunto nuevo, sustituir a un miembro nombrado en una colección, o crear un miembro si todavía no existe.
    - POST para crear un nuevo miembro en una colección.
    - PATCH para modificar elementos de un miembro o una colección existente.
    - DELETE para eliminar una colección o un miembro.
- Son sin estado. Es decir, la información no es mantenida por el servidor una vez que la transacción es completada.
- Utilizan un formato como XML o JSON para la estructura de representación de datos de paso de mensajes.

En las actividades que se proponen a continuación, se explorará e interactuará con las RESTful APIs por medio de la utilización de diferentes métodos y herramientas.

6.1.1 APIs con Postman

Postman, es una extensión gratuita para el navegador web Google Chrome que permite probar servicios web fácilmente, basta con indicar la URL, el método HTTP (POST, GET, etc.) y los parámetros de la petición [6].

Luego de haber añadido la extensión Postman, buscar en menú Inicio de Windows la carpeta "Aplicaciones de Chrome". Dentro de esta carpeta se encuentra la herramienta Postman.

1. Abrir Postman:
   Tomarse el tiempo para explorar y entender qué hace Postman y cómo se organiza.
   Postman es una herramienta muy potente y fácil de utilizar, sólo debe saberse algunas cosas para comenzar.
   Entonces, para realizar una llamada básica a una RESTful API, es necesario definir tres parámetros:
   - El Verbo HTTP (o método), los cuales fueron descritos anteriormente. Estos son: GET, PUT, POST, PATCH o DELETE.
     Hay muchos otros verbos HTTP disponibles pero los cinco mencionados son prevalentes en aplicaciones de automatización de red.
   - URI, quién define el recurso donde se actuará. Típicamente es un URL que apunta a una función específica, tipo de recurso o instancia de recurso en el servidor.
   - Las credenciales de autenticación. Se utilizará la autenticación básica con nombre de usuario "guest" y contraseña "guest".

   Una vez que los tres parámetros han sido definidos, la petición es enviada y el resultado a esta petición se muestra en la ventana principal de Postman.

   La figura 6-2, ilustra la ventana principal de Postman, resaltando los campos en donde se seleccionará el Método, se introducirá el URL, usuario y contraseña, además del lugar donde se obtendrán las respuestas a las peticiones.

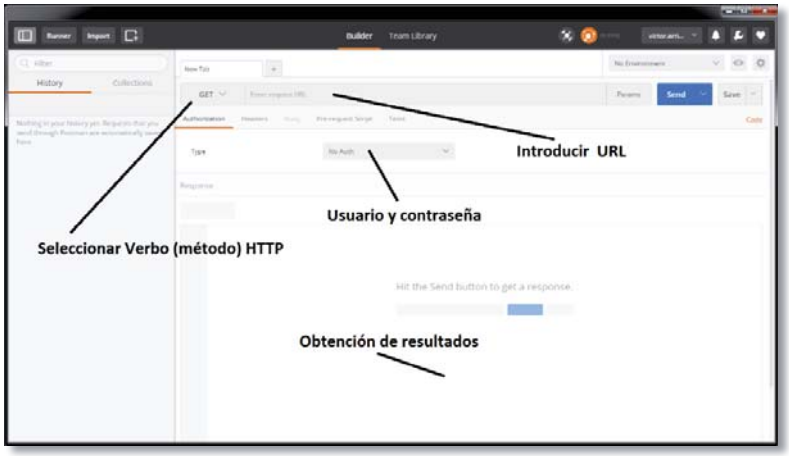


Figura 6-2. Ventana principal de Postman, extensión de Google Chrome. (Fuente: elaboración propia).

2. Seleccionar el Verbo GET.
3. Introducir el URL: http://**virl**:19399/
   Nota: remplazar las letras en negro "virl" del URL por la dirección IP que fue asignada al servidor VIRL. En este caso la dirección IP corresponde a 192.168.72.130 (http://192.168.72.130:19399/). Esta dirección puede encontrarse en el host VIRL (Máquina virtual) en el ícono "ip-address".
4. En el menú desplegable de autenticación, seleccionar "Basic Auth". Introducir "guest" como nombre de usuario y contraseña. Luego seleccionar "Update Request".
5. Seleccionar "Send" para enviar la petición al servidor VIRL.
   Hasta aquí se ha enviado una solicitud GET al motor de simulación VIRL, el que "vive" en el puerto TCP 19399. Esta solicitud ha causado que el motor de simulación envíe todas las RESTful APIs que están habilitadas para responder.
6. Dedicar tiempo para explorar y utilizar algunas de las APIs que pueden verse en la respuesta a la solicitud.
   Para esto, tener en consideración lo siguiente:
   - Hasta el momento no se han corrido simulaciones. Si se estuviera corriendo una simulación, muchas de las APIs que requieren <simulation-id>, <node-id>, o cualquier otro contenido mostrado como variable no funcionará.
   - Se puede seleccionar un URL de la lista de resultados, pero se debe restablecer las credenciales de autorización.
   - Puede utilizarse la lista del historial (a la izquierda) para volver a enviar la llamada API inicial y obtener una lista de las APIs de simulación.

   Algunas APIs "agradables e inofensivas" que se pueden explorar, remplazando como se muestra en el paréntesis (http://**virl**:19399/**XXXXX/YYYYY/ZZZZZ**), son:

   - /simengine/rest/list
   - /simengine/rest/licensing
   - /simengine/rest/subtypes
   - /simengine/rest/test
   - /openstack/rest/flavors
   - /openstack/rest/images
   - /openstack/rest/networks
   - /roster/rest/test

   Al explorar algunas de las APIs anteriores, no ignorar resultados que se obtiene demasiado rápido, hay mucha información importante contenida en estas respuestas.

6.1.2 APIs con cURL

cURL es un proyecto de software consistente en una biblioteca (libcurl) y un intérprete de comandos (curl) orientado a la transferencia de archivos. Este software permite enviar peticiones (hacer llamadas) API utilizando diferentes protocolos.

En el ejercicio planteado a continuación, se utilizará cURL para gestionar simulaciones.

cURL está disponible generalmente en sistemas Linux y Mac. En Windows se debe utilizar el software **Cygwin** (colección de herramientas open-sourse y GNU que provee de funcionalidad Linux). Este software puede descargarse en: https://www.cygwin.com

1. Una vez descargado e instalado el software Cygwin, abrir "Aygwin Terminal".
2. Se realizará una llamada API simple, donde se consultará el servicio AutoNetkit de VIRL. Introducir el comando 'curl -u guest:guest http://virl:19399/ank/rest/test'.
   Recordar que debe reemplazarse "**virl**" por la dirección IP asignada al servidor VIRL, tal como se ilustra en la figura 6-3. La respuesta a esta petición se muestra en la figura 6-3.

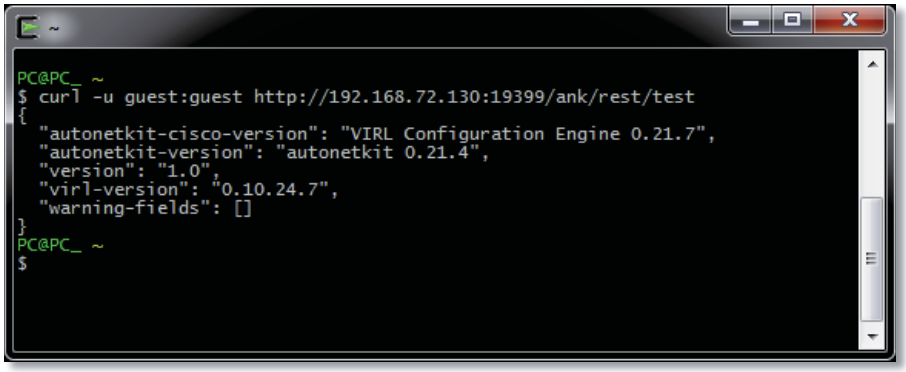


Figura 6-3. Consulta al servicio AutoNetkit de VIRL. (Fuente: elaboración propia).

Observar que la llamada realizada es muy similar a la que se hizo con Postman. Se tiene un URL, se provee con un parámetro de autenticación "**-u guest:guest**", pero no fue necesario especificar el Verbo o método. Esto es debido a que el verbo utilizado por defecto por cURL es GET. Para especificar un verbo es necesario utilizar "**-X**".

3. Se ocupará el mismo comando anterior pero se especificará el verbo PUT.
   'curl -X put -u guest:guest http://virl:19399/ank/rest/test'
   La respuesta a esta llamada se ilustra en la figura 6-4, donde puede apreciarse que no es posible acceder al servicio AutoNetkit.

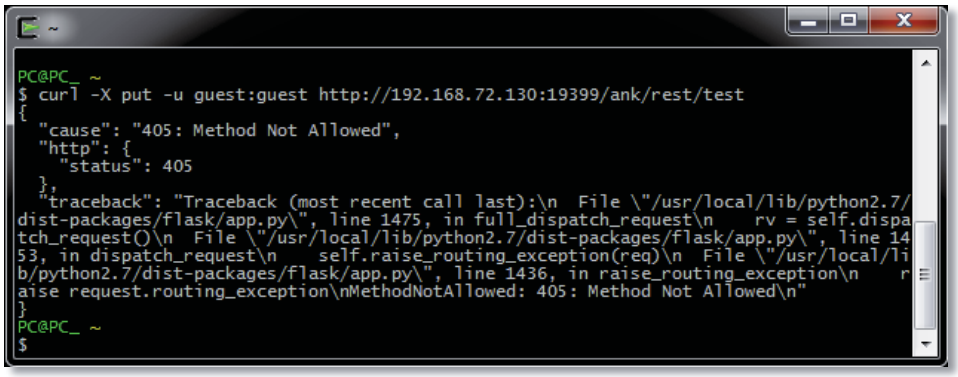


Figura 6-4. Respuesta a la llamada del servicio AutoNetkit de VIRL con el verbo PUT. (Fuente: elaboración propia).

El origen del error está en el mensaje "**405: Method not Allowed**". Esto indica que el método utilizado no corresponde. Puesto que todas las llamadas a RESTful API tienen un verbo asociado. En este caso, la petición realizada sólo está soportada mediante el verbo GET. Algunas llamadas API soportan más de un verbo y poseen diferentes acciones para cada uno. En este caso "**/simengine/rest/test**" sólo está soportado por el método GET.

Si se desea ver el total de la conversación entre cURL y las APIs VIRL, se debe incluir la opción "**-v**" en la línea de comando: 'curl **-v** -u guest:guest http://virl:19399/ank/rest/test' La respuesta al comando anterior se ilustra en la figura 6-5.

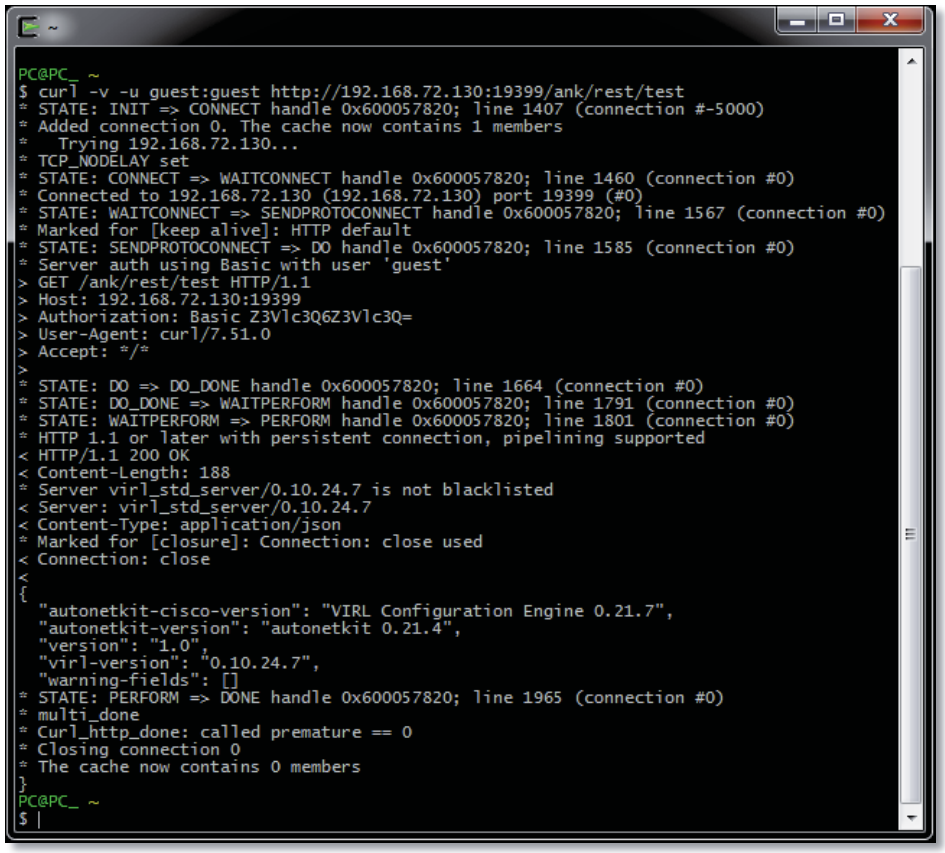


Figura 6-5. Conversación completa entre cURL y la consulta el servicio AutoNetkit. (Fuente: elaboración propia).

4. Se llamará al servicio UWM, en el puerto TCP 19400, para enumerar los proyectos OpenStack presentes en el host VIRL.
   Introducir el comando: 'curl -v -u guest:guest http://virl:19400/rest/projects'.
   UWM responde con una lista de todos los proyectos registrados, con toda la información que concierne a sus privilegios.
   La figura 6-6 ilustra la respuesta completa al comando anterior.

Figura 6-6. Respuesta del llamado al servicio UWM "/rest/projects", en el puerto TCP 19400. (Fuente: elaboración propia).

6.1.3 Detalles sobre APIs VIRL

Hasta el momento se han explorado 2 métodos para realizar llamadas API. Además se ha descubierto un par de cosas sobre las RESTful APIs:

- Requieren un URL.
- Cada API tiene un método o verbo específico asociado.

- Cada API devuelven un conjunto finito de código de estado que indica el éxito de la petición o algún tipo de fallo.

Obtener una lista de las APIs de la manera que se ha hecho hasta ahora puede ser útil, pero la información obtenida es parcialmente completa. Por ejemplo ¿Cómo podría saberse todos los posibles códigos de retorno? ¿Qué datos se necesitan para las operaciones PUT o POST?

VIRL provee un set completo de fácil acceso que contiene toda la información referente a las APIs. Esto se encuentra dentro del servicio UWM (ver figura 6-7).

1. Ir al servicio UWM. Recordar que el acceso se realiza en el navegador web, introduciendo la dirección IP que fue asignada al servidor VIRL. Luego ingresar a servicio "User Workspace Management" (UWM). Usuario y contraseña puede ser la misma utilizada anteriormente "guest". En el menú a la izquierda, seleccionar "Documentation".
2. Seleccionar STD API (ver figura 7-7). En la lista que se presenta, hacer clic en "simengine".

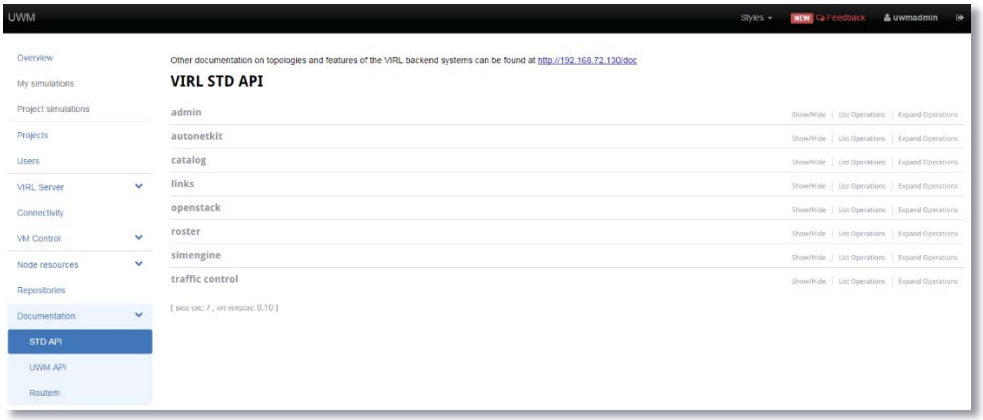


Figura 6-7. Vista del servicio UWM. (Fuente: elaboración propia).

Observar que en cada API disponible, puede verse:

- El verbo requerido.
- URL requerido, incluyendo todos los elementos variables.
- Una descripción de lo que realiza la API.

3. Localizar y abrir la API /simengine/rest/launch.

   Observar que la documentación completa de la API es mostrada, incluyendo:

   - Notas de implementación (*Implementation Notes*) que describen el uso y la acción de la API.
   - La "Clase de Respuesta" (*Response Class*). Que corresponde a "200 OK" o qué esperar cuando la llamada a la API es exitosa.
   - Los parámetros (*Parameters*) que pueden ser provistos por las llamadas API, incluyendo:
     - Nombre.

    - Valor esperado. Tener en cuenta que *{empty}* indica que el parámetro es opcional.
    - Descripción.
    - Tipo de parámetro.
    - Tipo de dato: string, booleano, entero, etc.
  - Las posibles respuestas (*Response Messages*), incluyendo:
    - Códigos de respuesta.
    - Razones de respuesta.
    - Esquema de modelos de respuesta. Esto es el formato de los datos que serán devueltos con la respuesta.
4. Tomarse el tiempo necesario para explorar las otras APIs que están incluidas en SimEngine y en otros servicios VIRL.

6.1.4 Automatización de funciones

En los ejercicios anteriores, se realizaron llamadas API, logrando obtener respuestas a estas peticiones. Además, se vio que VIRL provee de un set completo que contiene toda la información referente a las APIs VIRL. Pero si se quisiera automatizar alguna acción como lanzar una simulación, detener nodos. ¿Cómo podría realizarse esto?

A continuación se explicará cómo es posible lanzar una simulación utilizando cURL y cómo utilizando Python puede detenerse uno o varios nodos. La revisión de estos tópicos es sólo para demostrar que es posible automatizar funciones en VIRL. No tendría sentido revisar estos temas con más profundidad, puesto que el objetivo de este informe no considera realizar algún tipo de programación, basta con conocer las capacidades.

cURL permite lanzar simulaciones por medio de línea de comando. El siguiente comando es utilizado para realizar esta función:

‘curl -X POST -u guest:guest --header "Content-Type:text/xml;charset=UTF-8" --data-binary @API.virl http://virl:19399/simengine/rest/launch?session=a1b2c3’

El comando anterior, es analizado en la tabla 6-1.

Tabla 6-1. Elementos y sus respectivas funciones para lanzar una simulación mediante cURL.

Elemento	Uso
-X POST	Utilización del verbo POST
-u guest:guest'	Utilización de autenticación básica. Con nombre y contraseña: **guest**
--header "Content-Type:text/xml;charset=UTF-8"	Los datos incluidos son texto XML y codificación UTF-8.
--data-binary @API.virl	Enviar el archivo API.virl. Corresponde al nombre del archivo que contiene la topología que se lanzará.
http://virl:19399/simengine/rest/launch	Llamado al servicio /rest/launch de la API SimEngine.
?session=a1b2c3	Envía el parámetro opcional **session** con el valor **a1b2c3**. Esto indica el nombre que tendrá la simulación lanzada (a1b2c3).

1. Ir a VM Maestro. Crear una nueva topología utilizando 2 nodos IOSv, conectarlos entre sí y generar configuraciones. Nombrar al archivo “API.virl”.
   Cerrar la topología recién creada.
2. Ir a la carpeta “My Topologies” donde se alojó el archivo “API.virl”.
3. Mover el archivo “API.virl” a la carpeta “PC” en la dirección “C:\cygwin64\home\PC”.
4. Abrir la venta “Cygwin64 Terminal” e introducir el siguiente comando:
   ‘curl -X POST -u guest:guest --header "Content-Type:text/xml;charset=UTF-8" --data-binary @API.virl http://virl:19399/simengine/rest/launch?session=a1b2c3’
   *No olvidar cambiar “**virl**” por la dirección IP asignada al servidor VIRL.
5. Volver a VM Maestro y observar que se ha lanzado una simulación con el nombre “**a1b2c3**”.
   Clic derecho en “**a1b2c3**” y seleccionar ver simulación (*View Simulation*) para observar la topología.

Entonces, mediante la utilización de cURL, es posible lanzar simulaciones de archivos locales de topología. Puede visualizarse que lo anterior podría convertirse en una serie de comandos utilizados para un proceso de automatización. Sin embargo, el mecanismo más potente para el uso de las APIs es por medio de Python.

Python posee una biblioteca llamada “requests”, quien provee un amplio conjunto de funciones de llamada que permiten las comunicaciones con los servicios basados en la web, incluyendo RESTful APIs.

En el siguiente ejercicio se utilizará un script Python llamado “stop-node.py” para detener un nodo en la simulación que fue lanzada con cURL.

El script se adjunta en la siguiente página, sólo se necesita darle la extensión “.py” para continuar con el ejercicio.

```
#!/usr/bin/env python

# Import the required libraries, most notably for our
# purposes 'requests' and 'logging'

import os, requests, sys, logging

# Enable debugging in httplib so we see the request and response
headers

try:
import http.client as http_client
except ImportError:
import httplib as http_client

http_client.HTTPConnection.debuglevel = 1

# Initialize the logging function data structures

logging.basicConfig()
logging.getLogger().setLevel(logging.DEBUG)
requests_log = logging.getLogger("requests.packages.urllib3")
requests_log.setLevel(logging.DEBUG)
requests_log.propagate = True

# Define our function to stop one or all nodes in the simulation

def stop_node(simulation, node):
virl_host = "virl"
username  = password = "guest"
url       = "http://%s:19399/simengine/rest/update/%s/stop" %
(virl_host, simulation)
headers   = {'content-type': 'text/xml'}
payload   = {'nodes': node}


result = requests.put(url, auth=(username, password),
params=payload, headers=headers)

def main():
if len(sys.argv) != 3:
sys.stdout.write(str(sys.argv[0]))
print ":  usage: stop-node.py <simulation-name> <node-name>"
return 1
else:
stop_node(str(sys.argv[1]).strip(),str(sys.argv[2]).strip())
return 0


if __name__ == '__main__':
sys.exit(main())
```

Fuente: http://virl-dev-innovate.cisco.com/dl/stop-node.py

Los bits más relevantes del script “stop-node” son:

```
#!/usr/bin/env python

# Import the required libraries, most notably for our
# purposes 'requests' and 'logging'

import os, requests, sys, logging
```

La primera línea le indica al sistema operativo que debe interpretar lo que sigue como código Python.

El comando “import” carga las librerías necesarias para el script. Específicamente, las librerías “requests” y “logging”. La primera es utilizada para realizar las llamadas a las RESTful APIs y la segunda permite el registro detallado de las solicitudes y respuestas de las APIs.

Luego, en la siguiente sección se define una función que realizará las llamadas API, junto con los parámetros, contantes y variables necesarias.

```
# Define our function to stop one or all nodes in the simulation

def stop_node(simulation, node):
virl_host = "virl"
username  = password = "guest"
url       = "http://%s:19399/simengine/rest/update/%s/stop" %
(virl_host, simulation)
headers   = {'content-type': 'text/xml'}
payload   = {'nodes': node}
```

La función que se define acepta el ingreso de dos variables desde la línea de comandos. Estas variables describen la simulación que se desea manipular y el nodo que se quiere detener.

Se definen constantes para “virl_host”, “username”, “password” y “headers”. El script podría ser adaptado para introducir estas constantes por medio de la línea de comandos.

Debe observarse que la constante asignada a “url” contiene el nombre de la API y el servicio al que se realizará la llamada (/simengine/rest/update/<simulation>/stop). Se debe ingresar la dirección IP que fue asignada al servidor VIRL tal como se ha realizado antes, sólo que acá se muestra como una variable (%s) en vez de “virl”.

Finalmente la variable “payload” representa el parámetro que se desea transmitir en conjunto con la llamada API, que en este caso especifica el nodo que se detendrá.

El último elemento importante en el script es:

```
result    =    requests.put(url,    auth=(username,    password),
params=payload, headers=headers)
```

Esta línea realiza efectivamente la llamada API al host VIRL, llamando a la función “put” de la librería “requests”

Esta línea pasa a lo largo de los elementos que fueron definidos, incluyendo URL, autenticación, parámetros y los datos de encabezado. Los resultados de las llamadas se almacenan en la variable denominada “result” para su posterior visualización.

El resto del contenido del script proporciona lo necesario para la estructura de datos en la librería “logging”, además del paso de argumentos desde la línea de comandos a la función “stop_node”.

1. Una vez que el script posee la extensión “.py”, moverlo a la carpeta “PC” en la dirección “C:\cygwin64\home\PC”.
   Abrir la ventana “Cygwin64 Terminal” e introducir el comando ‘ls’ (ele ese) para asegurarse de que el script aparece dentro de la lista.
   Recordar que se debe cambiar la variable “%s” por la dirección IP que fue asignada al servidor VIRL.
2. Ejecutar el script “stop-node.py” para detener el nodo “iosv-1”.
   Observar que en la respuesta aparece el URL completo, el que es formado y enviado por la función “requests.put”. Junto con el resultado viene un mensaje con “200 OK” el que indica que la llamada a la API se autenticó, se formateó correctamente y se ejecutó. Por lo anterior, es válido obtener esta misma respuesta (“200 OK”) para un nodo que no existe en la topología, puesto que la llamada puede ser autenticada, formateada correctamente y ejecutada sin la necesidad de que el nombre del nodo exista en la topología.
3. Volver a VM Maestro y observar que el nodo “iosv-1” ha sido detenido.

Resumen

Se conoció que VIRL depende de RESTful APIs para su funcionamiento. De hecho, las APIs siempre están presentes y son utilizadas por cada elemento principal de VIRL. Fue utilizada la herramienta Postman (extensión del navegador web Google Chrome) para explorar algunos de estos servicios mediante la realización de llamadas API y el análisis de sus respuestas.

Fue utilizada la herramienta cURL para realizar llamadas API mediante la introducción de comandos, a diferencia de cuando se realizó con Postman, donde bastó con seleccionar algunos objetos en los menús desplegables de la extensión.

Se exploró el servicio UWM (User Workspace Management), donde puede encontrarse toda la información referente a las APIs VIRL. Es posible encontrar una descripción de la función que realiza cada API, los verbos o métodos que utilizan cada API, la clase de respuesta, entre otros.

Posteriormente fue utilizada la herramienta cURL para lanzar una simulación mediante la llamada a los servicios respectivos para realizar esta acción. También se vio que la herramienta Python es la más idónea para realizar automatizaciones en las simulaciones debido a la disponibilidad de bibliotecas que permiten un alto manejo en servicios basados en la web como las RESTful APIs. Donde en esta oportunidad fue detenido un nodo en la simulación que fue lanzada por cURL.

6.2 Características de simulación

Al aprovechar al máximo el entorno virtual, podrá notarse que las características avanzadas que ofrece VIRL no se encuentran disponibles en muchas redes reales. En este bloque se conocerán aspectos sobre el comportamiento de la red, donde será posible seguir el flujo de los paquetes de datos, introducir latencia, perdida de paquetes. Todo esto con el objetivo de realizar pruebas bajo ciertas condiciones introducidas a la red simulada. [30].

6.2.1 Captura de Paquetes con Wireshark

La captura de paquetes es uno de los métodos más comunes y eficaces que los ingenieros de redes y sistemas aprovechan para resolver problemas, ya que esto les permite ver lo que está sucediendo en el cable. Dentro de una simulación virtual, los paquetes fluyen como lo harían en cualquier red física. VIRL proporciona la capacidad de capturar paquetes en cualquier interfaz de red de un nodo. Es posible guardar los datos capturados en un archivo ".pcap" y analizarlo utilizando Wireshark, o también realizar captura en vivo para ver el flujo de paquetes a medida que viajan a través de una interfaz de red.

Antes de continuar, se debe instalar el software Wireshark. Wireshark es una herramienta gratuita para visualizar y analizar los paquetes capturados. La otra herramienta que se necesitará para realizar capturas en vivo es NetCat, el cual se conectará al socket de escucha en el servidor VIRL mientras realiza capturas en vivo. En Windows se puede descargar un ejecutable de NetCat ("nc.exe") del sitio web: https://eternallybored.org/misc/netcat/

Para iniciar una captura de paquetes, primero debe especificarse la interfaz de red y el nodo donde se realizará la captura. Debe tenerse una simulación corriendo para hacer esto. El archivo ".pcap" capturado se almacena en el servidor VIRL para descargarlo.

1. Lanzar la simulación e ingresar a UWM.
   Después de haber iniciado la simulación en VM Maestro, se debe iniciar sesión en UWM utilizando la misma cuenta que se utilizó para iniciar la simulación. Por defecto la contraseña y usuario es “guest”. Si ha iniciado sesión utilizando el usuario "uwmadmin", no será posible manipular simulaciones “privadas” y proyectos “privados” que no se hayan iniciado utilizando esta cuenta.
2. Seleccionar la interfaz e iniciar la captura de tráfico.
   Clic en "MY simulations" en el menú de la izquierda, abrir la simulación y buscar "interfaces". Aquí podrán verse todas las interfaces de red de un nodo en ejecución. Seleccione una o más interfaces en las que desee realizar la captura de paquetes haciendo clic donde se indica en la figura 6-8.

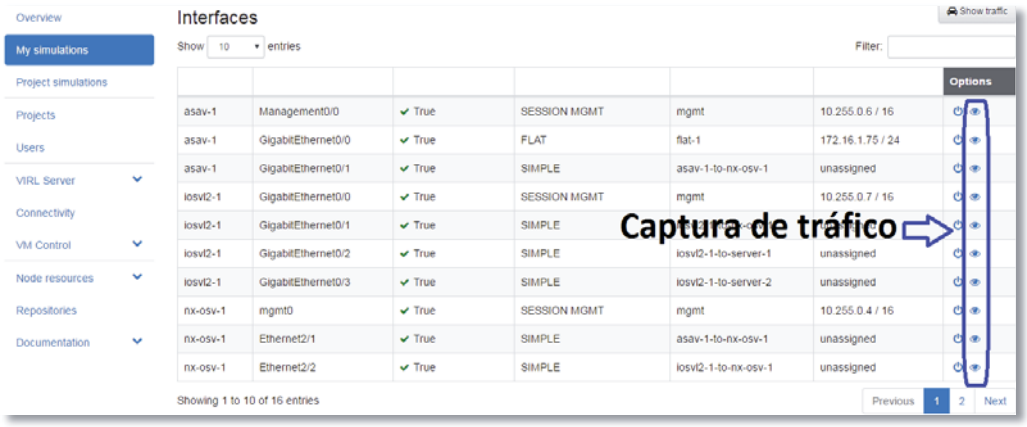


Figura 6-8. Selección de interfaz para captura de tráfico. (Fuente: elaboración propia).

Seleccionar el modo de captura que se indica en la figura 6-9.

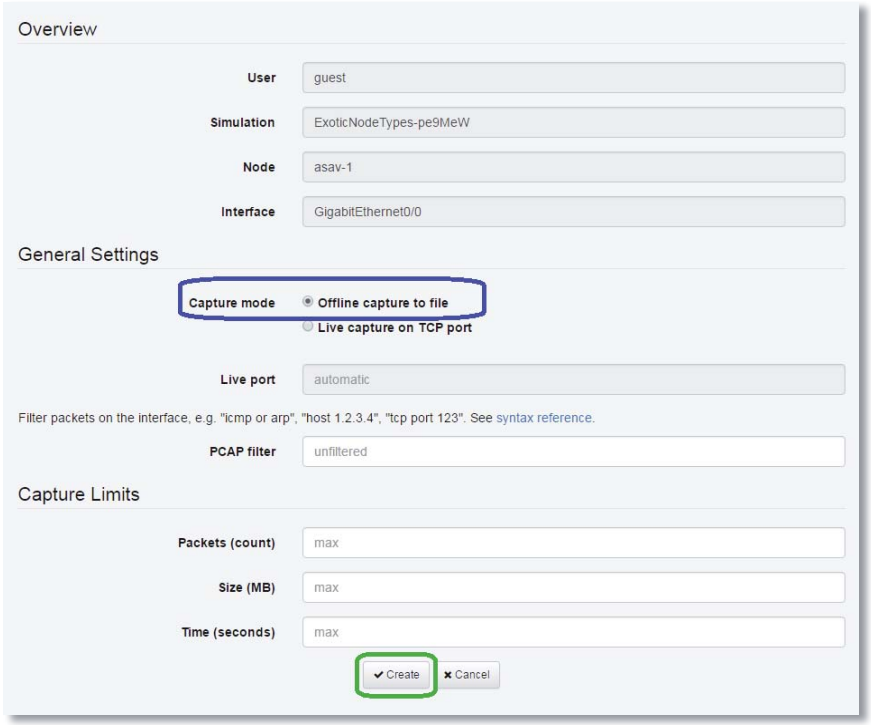


Figura 6-9. Selección del modo de captura. (Fuente: elaboración propia).

De la figura 6-9, puede especificarse el filtro PCAP, el máximo de paquetes, el tamaño y el tiempo por el cual se desea realizar la capturar. Los valores predeterminados son: no PCAP (capturar todo), máximo 1.000.000 de paquetes, 10MB y 86.400 segundos. Haga clic en "Create" para configurar el perfil de captura.

3. Descargar el archivo “.pcap” y abrirlo en Wireshark.

Una vez creada la captura. Ir a “Traffic captures”, hacer clic en la figura con forma de ojo para descargar el archivo. Doble clic para abrir el archivo.
La figura 6-10, ilustra un archivo “.pcpa” abierto con Wireshark.

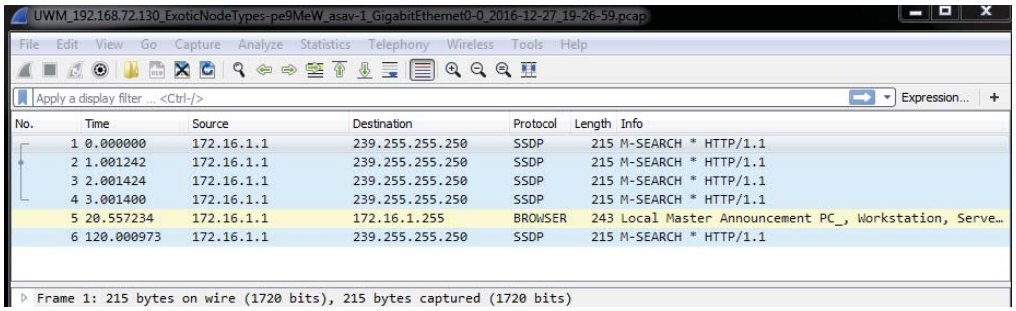


Figura 6-10. Archivo “.pcpa” abierto con Wireshark. (Fuente: elaboración propia).

6.2.2 Captura de paquetes en vivo

La idea detrás de la captura de paquetes en vivo, es que puede verse acciones en vivo, como paquetes que fluyen dentro y fuera de la interfaz de un nodo. Esta opción abre un puerto de escucha en el servidor VIRL, lo que permite que una aplicación rastreadora (sniffer) de paquetes externa (por ejemplo, Wireshark) se conecte y muestre el flujo de paquetes a medida que sucede. Como se está viendo la captura de paquetes en el PC, no se almacenan datos en el servidor VIRL. Recordar que es necesario el ejecutable “nc.exe”.

1. Seleccionar la Interfaz e inicializar la captura en vivo de paquetes.

   Seleccionar "Live capture on TCP port" y especificar un puerto entre 10.000 y 17.000. Este es el puerto al que el servidor VIRL enviará las capturas en vivo. Se utilizará el puerto 12.000 en este ejemplo (ver figura 6-11). Iniciar la captura de paquetes.

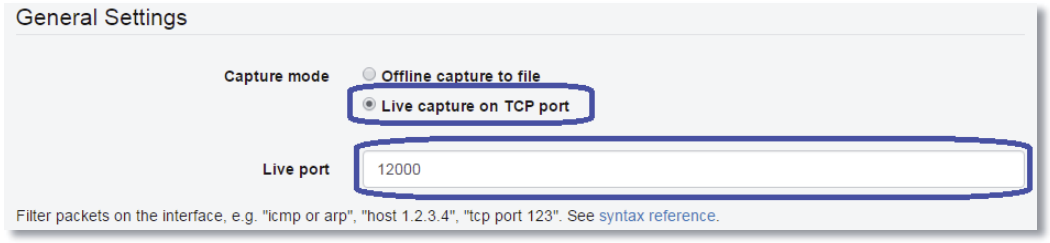


Figura 6-11. Selección del modo de captura y el puerto para enviar la captura en vivo. (Fuente: elaboración propia).

2. Configurar un "conducto" en Wireshark para detectar los paquetes capturados.
   Abrir Wireshark e ir al menú "Capture”, “Options" y luego "Manage Interfaces".
   Especificar un archivo temporal en el PC para usarlo como un "conducto" que sirva como túnel a los paquetes capturados en vivo. Debe especificarse un archivo en el directorio /tmp. El nombre del archivo puede ser arbitrario pero debe tener la extensión ".fifo" para

que Wireshark lo reconozca como un "conducto". Por ejemplo "/tmp/remotecapture.fifo" (ver figura 6-12).
Iniciar la captura de paquetes con Wireshark.

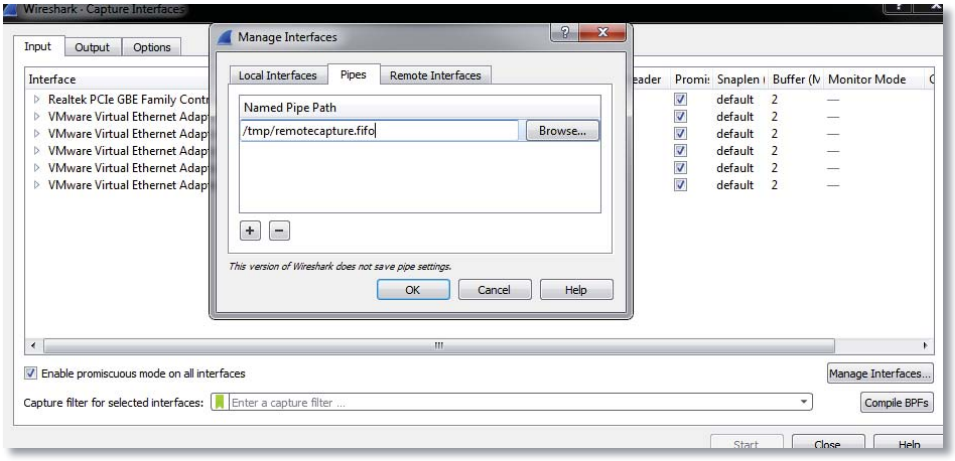


Figura 6-12. Especificación del archivo en el directorio /tmp. (Archivo “conducto”). (Fuente: elaboración propia).

Ahora Wireshark está captando cualquier paquete entrante al archivo “conducto”. El siguiente paso es usar el ejecutable "nc.exe" como *sniffer* en el puerto del servidor VIRL.

3. Utilizar NetCat como *Sniffer* y enviar los paquetes a Wireshark.
   En el punto 1, se estableció una captura de paquetes, donde el servidor VIRL envió todos los paquetes capturados al puerto TCP 12.000. Utilizando la herramienta NetCat, es posible conectarse al puerto TCP 12.000 en VIRL y recoger y almacenar todos los paquetes en un archivo temporal.
   Abrir “nc.exe” e introducir el siguiente comando para realizar lo planteado.
   ‘nc 192.168.72.130 12000 > /tmp/remotecapture.fifo’

   Cuando el buffer está lleno, la captura funciona de manera FIFO (First-In-First-Out).

6.2.3 Introducción de Latencia, Jitter y Pérdida de Paquetes

En las redes del mundo real, la latencia, el *jitter* y la pérdida de paquetes son inevitables. Con el fin de hacer la simulación lo más cercana posible al mundo real, VIRL permite agregar latencia personalizada, jitter y pérdida de paquetes en cualquier enlace que se conecte a un nodo. Esta característica es muy útil para las grandes empresas y proveedores de servicios de Internet (ISP) para diseñar y probar *Optimized Edge Routing* (OER), quien hace ajustes adaptativos y dinámicos de enrutamiento basados en criterios como tiempo de respuesta, pérdida de paquetes, jitter, la disponibilidad de rutas, la distribución de carga de tráfico y las políticas de minimización de costos.

Existen tres formas de configurar estos parámetros: VM Maestro, UWM y en “Live Visualization”. Debe tenerse en consideración que los valores personalizados sólo tienen efecto durante la ejecución de la simulación, puesto que no se guardan en el archivo de topología.

VM Maestro

La primera opción para configurar estos parámetros es mediante VM Maestro. Para realizar la configuración, se debe hacer clic derecho en el enlace que se quiere modificar y luego ir a “Modify link parameters…”. La figura 6-13 ilustra lo anterior.

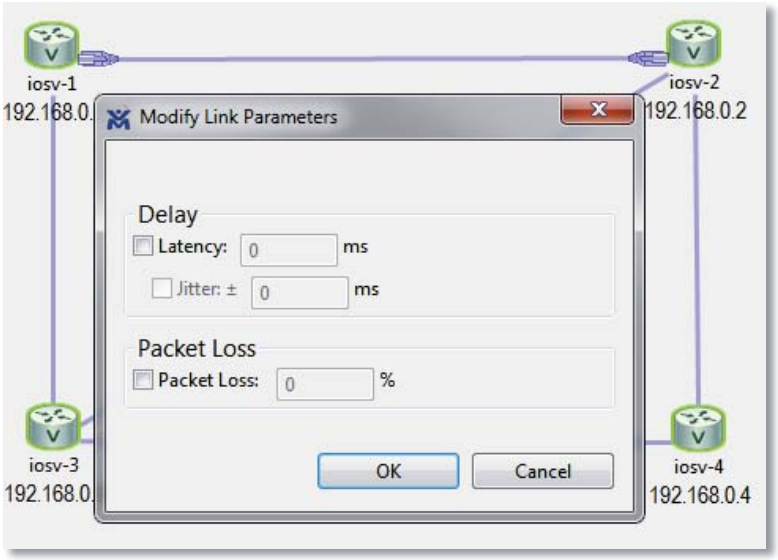


Figura 6-13. Introducción de latencia, jitter y pérdida de paquetes en VM Maestro. (Fuente: elaboración propia).

User Workspace Management

La segunda alternativa para editar los parámetros de enlace es mediante UWM. Para esto se debe ingresar a UWM utilizando el mismo usuario y contraseña con la que se inicializó la simulación. Por defecto es “**guest**” para usuario y contraseña.

Ir a “My simulations”, abrir la simulación que se desea modificar y luego buscar la sección “Links”. En la pestaña “Options” se realiza la modificación de estos parámetros para cada enlace existente en la simulación. Ver figura 6-14.

Figura 6-14. Introducción de latencia, jitter y pérdida de paquetes en UWM. (Fuente: elaboración propia).

Live Visualization

Por último, la tercera alternativa para agregar latencia, jitter y/o pérdida de paquetes, es en la pestaña "Live Visualization". Ver figura 6-15.

Al hacer clic en "Live Visualization", se abrirá una nueva pestaña en el navegador web, la que mostrará la visualización en vivo de la simulación. Para modificar los parámetros se debe hacer clic en el enlace que se desee. Aparecerá una ventana donde se debe introducir los valores de los parámetros.

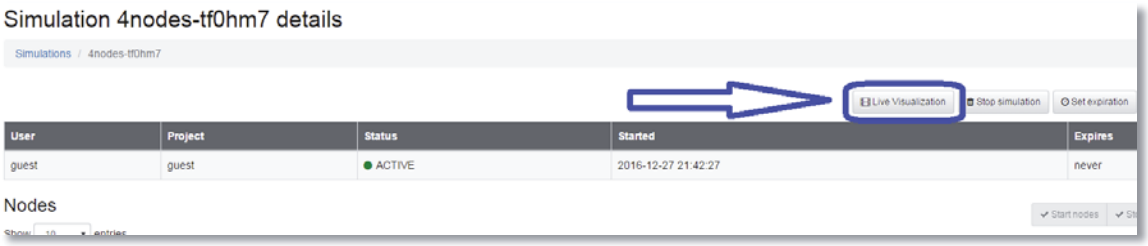


Figura 6-15. Enlace para obtener la vista "Live Visualization" de la simulación. (Fuente: elaboración propia).

6.2.4 Inyección de Rutas Artificiales

Cuando se trabaja con BGP en un entorno de laboratorio, a menudo es necesario generar algunas rutas para estudiar el filtrado de rutas y comportamientos de AS_path. Tradicionalmente había dos maneras de hacer esto. La primera es creando un grupo de interfaces Loopback y luego importar las rutas conectadas en el interior de BGP. La otra forma es escribir un script TCL para crear repetidamente algunas rutas estáticas o interfaces Loopback [30]. ¿Qué pasaría si se necesitan miles de rutas en la tabla de enrutamiento? Realizar esto con "fuerza bruta" o semi-manual podría no ser posible. O si se pudiera, la configuración del Router sería demasiado larga para ser utilizada de manera práctica.

Routem es una aplicación generadora de tráfico en control-plane de Cisco. Está embebido en el nodo 'lxc-routem' (ver figura 6-16). Puede utilizarse para generar rutas o tráfico para ISIS, OSPF, MSDP, RIP, ICMP, etc. En UWM-Documentations-Routem, puede encontrarse la documentación completa de esta aplicación, además de configuraciones de muestra. Mediante esta aplicación es

posible generar rutas BGP para realizar estudios de filtrado de rutas o comportamientos de AS_path.

No se expondrán ejemplos al respecto, ya que este tema se aleja un poco de los objetivos. Por tanto, basta con hacer referencia a que es posible inyectar rutas BGP artificiales mediante la aplicación Routem. La figura 6-16, resalta al nodo “lxc-routem”, el que se encuentra en la paleta de nodos de VM Maestro.

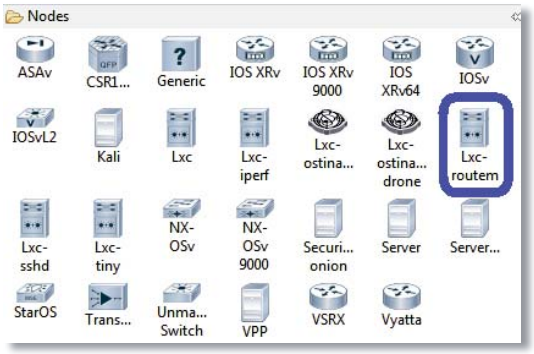


Figura 6-16. Nodo Lxc-routem, donde se encuentra embebida la aplicación Routem. (Fuente: elaboración propia).

6.2.5 Visual Traceroute

Mostrar la ruta que siguen los datos es una de las mejores características que ofrece VIRL. Se debe especificar el nodo inicial y final para enviar paquetes de datos y rastrear la ruta que siguen. Esto se consigue a través de un efecto animado que muestra cómo los paquetes fluyen a través de la red. Para ocupar esta herramienta, se debe ingresar a UWM-“My simulations” y seleccionar la simulación en la que se desea utilizar Visual Traceroute. Clic en el botón "Live Visualization" (ver figura 6-15), se abrirá una nueva pestaña con la visualización en vivo de la simulación.

Hacer clic en el nodo donde se desea iniciar el tráfico y seleccionar “Trace From”. Luego hacer clic en el nodo destino y seleccionar “Trace To”. La figura 6-17 ilustra lo anterior.

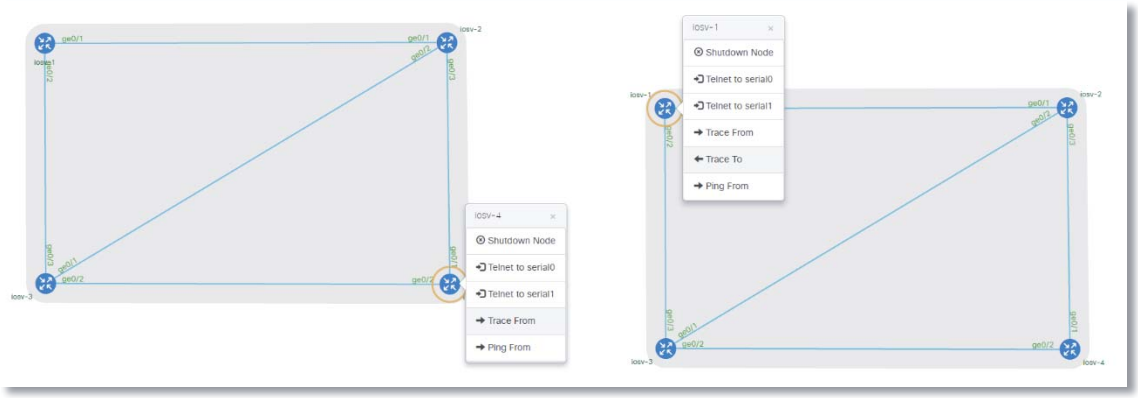


Figura 6-17. Selección del nodo inicio y destino para los paquetes de datos. (Fuente: elaboración propia).

El trazo de la ruta que siguen los datos aparece a continuación, mostrando una animación que comienza desde el nodo inicio hasta el nodo destino, tal como se muestra en la figura 6-18.

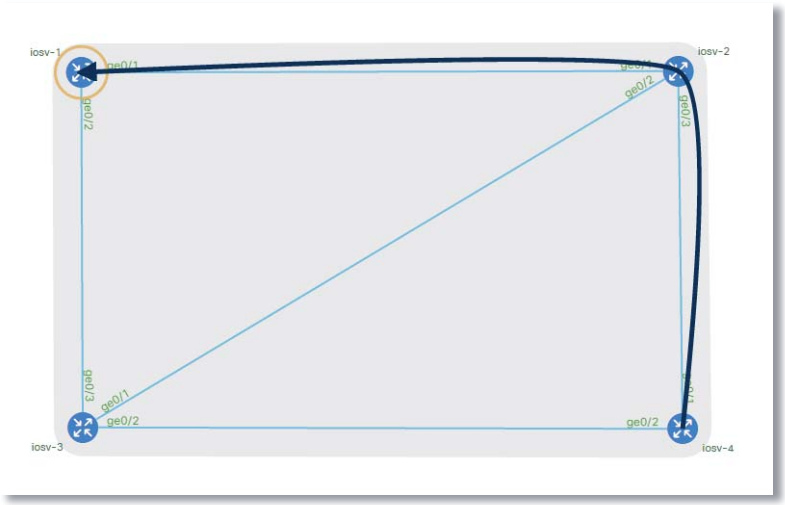


Figura 6-18. Trazo de la ruta que siguen los paquetes de datos. (Fuente: elaboración propia).

Para realizar pruebas, puede deshabilitarse la interfaz de un nodo. En la figura 6-19, se ha deshabilitado la interfaz Gig0/1 del nodo iosv-1, observar que se encuentra en color rojo a diferencia de las demás interfaces. Esto se realiza haciendo clic en el nombre de la interfaz y luego en deshabilitar. Cuando la tabla de enrutamiento se ha actualizado, la ruta de los paquetes de datos es trazada nuevamente, tomando otro camino para llegar al nodo destino, tal como se ilustra en la figura 6-19.

Figura 6-19. Nueva ruta de los paquetes luego de deshabilitar la interfaz Gig0/1 del nodo iosv-1. (Fuente: elaboración propia).

6.2.6 Generación de Trafico con Ostinato

En una red de laboratorio, a menudo no se tiene suficiente tráfico como para simular el que posee una red real. Para simular una red moderadamente cargada y observar los comportamientos de QoS (calidad de servicio) y enrutamiento, se necesita una herramienta artificial de generación de tráfico.

Ostinato es un generador y analizador de tráfico de red de código abierto con una interfaz gráfica de usuario, posee una potente API Python para pruebas y automatización de redes. Ostinato puede enviar paquetes de varios streams con diferentes protocolos, a diferentes velocidades y diferentes destinos. Cisco ha habilitado Ostinato en VIRL como un LXC "liviano".

Para utilizar Ostinato debe descargarse desde el sitio web "http://ostinato.org/". Éste realizará llamadas al motor "drone" el cual se encuentra disponible en la paleta de nodos de VM Maestro (ver figura 6-20).

Figura 6-20. Motor "drone" utilizado por Ostinato en VIRL. (Fuente: elaboración propia).

Cómo utilizar Ostinato, se escapa de los objetivos de este informe. Por tanto, para el propósito, es suficiente saber que es posible generar tráfico para simular carga de red. Sin embargo, en la página web "http://ostinato.org/docs/", se encuentra información disponible para aprender a utilizar este software.

6.2.7 Simulación Test de Velocidad con iPerf

iPerf es una popular herramienta de red que fue desarrollada para medir el desempeño del ancho de banda para TCP y UDP en una red cableada o inalámbrica. Al ajustar varios parámetros y características de los protocolos TCP y UDP, las pruebas pueden proporcionar una visión de la disponibilidad de ancho de banda de la red, delay, jitter y pérdida de datos. iPerf debe descargarse desde la página web "https://iperf.fr/iperf-download.php".

En VIRL viene incluido un servidor iPerf-LXC. Para demostrar cómo usar iPerf para medir el rendimiento de la red, se construirá una topología simple que consta de tres routers IOSv y dos servidores iPerf. Se realizará la configuración inicial con AutoNetkit.

La figura 6-21 ilustra la simulación de la topología.

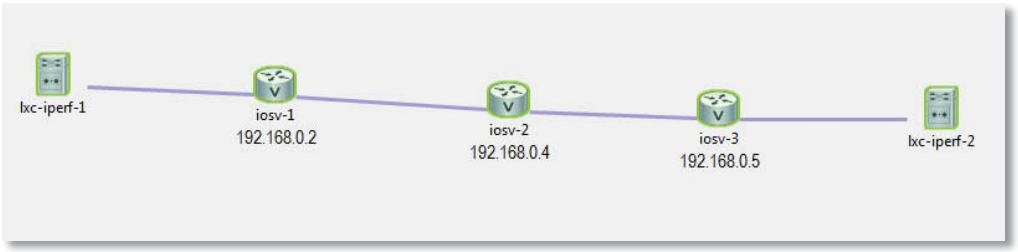


Figura 6-21.Simulación de topología con servidor y cliente iPerf. (Fuente: elaboración propia).

iPerf es una aplicación cliente-servidor. Entonces, para medir el rendimiento entre los puntos A y B, debe ejecutarse en el punto B el servidor iPerf, y en el punto A el cliente iPerf. El cliente lanza un montón de paquetes al servidor y mide la velocidad de transmisión de los datos a través de la red.

1. Lanzar el servidor iPerf en el nodo “lxc-iperf-2”.
   Ingresar al nodo “lxc-iperf-2”. La contraseña es “cisco”.
   Iniciar el servidor iPerf mediante el siguiente comando:
   ‘iperf –s’
   La figura 6-22, ilustra la respuesta al comando donde se aprecia el puerto por donde escucha el servidor y el tamaño de la ventana TCP.

```
cisco@lxc-iperf-2$ iperf -s
bind failed: Address already in use
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
```

Figura 6-22. Respuesta del nodo “lxc-iperf-2” al comando “iperf –s”. (Fuente: elaboración propia).

2. Iniciar el test en el nodo “lxc.iperf-1”.
   Se utilizará el nodo “lxc-iperf-1” como cliente iPerf para realizar el test.
   Ingresar al nodo “lxc-iperf-1” e inicializarlo como cliente iPerf introduciendo el comando ‘iperf –c’. Además se debe apuntar el tráfico hacia la dirección IP del servidor “lxc-iperf-2”. En conjunto con lo anterior, se introducirán los parámetros de prueba, los cuales son:
   - **–P** se refiere al número de streams TCP paralelos.
   - **–w** se refiere al tamaño de la ventana TCP.

   La figura 6-23, ilustra la respuesta a lo indicado anteriormente.

```
cisco@lxc-iperf-1$ iperf -c 10.0.0.18 -P10 -w1000k
------------------------------------------------------------
Client connecting to 10.0.0.18, TCP port 5001
TCP window size:   416 KByte (WARNING: requested 1000 KByte)
------------------------------------------------------------
[  8] local 10.0.0.6 port 54166 connected with 10.0.0.18 port 5001
[  3] local 10.0.0.6 port 54161 connected with 10.0.0.18 port 5001
[  6] local 10.0.0.6 port 54164 connected with 10.0.0.18 port 5001
[  4] local 10.0.0.6 port 54162 connected with 10.0.0.18 port 5001
[  7] local 10.0.0.6 port 54165 connected with 10.0.0.18 port 5001
[ 10] local 10.0.0.6 port 54167 connected with 10.0.0.18 port 5001
[  9] local 10.0.0.6 port 54168 connected with 10.0.0.18 port 5001
[ 12] local 10.0.0.6 port 54170 connected with 10.0.0.18 port 5001
[  5] local 10.0.0.6 port 54163 connected with 10.0.0.18 port 5001
[ 11] local 10.0.0.6 port 54169 connected with 10.0.0.18 port 5001
[  6]  0.0- 9.1 sec    544 KBytes    491 Kbits/sec
[  9]  0.0-10.5 sec    584 KBytes    454 Kbits/sec
[  7]  0.0-10.7 sec    584 KBytes    445 Kbits/sec
[ 11]  0.0-10.8 sec    584 KBytes    443 Kbits/sec
[  3]  0.0-11.4 sec    584 KBytes    419 Kbits/sec
[ 10]  0.0-12.1 sec    544 KBytes    369 Kbits/sec
[  8]  0.0-12.3 sec    544 KBytes    362 Kbits/sec
[  4]  0.0-13.5 sec    544 KBytes    329 Kbits/sec
[  5]  0.0-14.7 sec    736 KBytes    411 Kbits/sec
[ 12]  0.0-15.7 sec    544 KBytes    284 Kbits/sec
[SUM]  0.0-15.7 sec  5.66 MBytes  3.03 Mbits/sec
```

Figura 6-23. Test de velocidad. (Fuente: elaboración propia).

Se enviaron 10 streams paralelos con una ventana TCP de 1000[kB] hacia el servidor iPerf (lxc-perf-2). En la prueba, fueron transmitidos 5.66[MB] en 15.7[s]. El rendimiento fue de 3.03[Mbits/s] (ver figura 6-23).

Para realizar una prueba, se modifica el parámetro “Packet Loss” del enlace que une a los nodos iosv-1 e iosv-2 (ver figura 6-24). Se introduce un 20% de pérdida de paquetes y se vuelve a realizar la misma prueba. La figura 6-25, muestra los resultados de iPerf, los que confirman que la prueba tardó mucho más tiempo en completarse, debido a que al menos un 20% de paquetes tuvieron que ser retransmitidos. Esto afectó la eficiencia general de la red en mucho más del 20%.

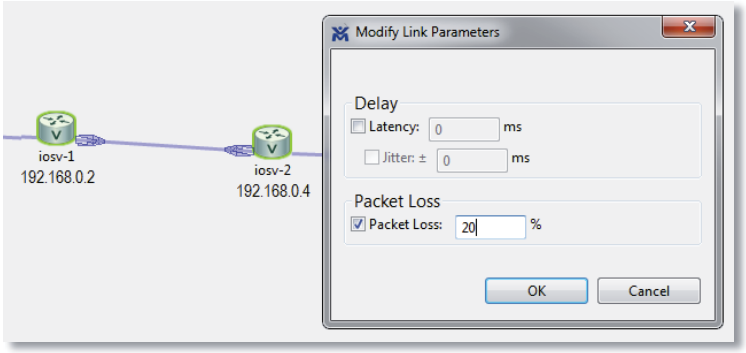


Figura 6-24. Se modifica el parámetro “Packet Loss” en un 20% para el enlace que une a los nodos iosv-1 e iosv-2. (Fuente: elaboración propia).

```
cisco@lxc-iperf-1$ iperf -c 10.0.0.18 -P10 -w1000k
------------------------------------------------------------
Client connecting to 10.0.0.18, TCP port 5001
TCP window size:   416 KByte (WARNING: requested 1000 KByte)
------------------------------------------------------------
[  4] local 10.0.0.6 port 54541 connected with 10.0.0.18 port 5001
[  3] local 10.0.0.6 port 54540 connected with 10.0.0.18 port 5001
[  6] local 10.0.0.6 port 54543 connected with 10.0.0.18 port 5001
[  7] local 10.0.0.6 port 54544 connected with 10.0.0.18 port 5001
[  8] local 10.0.0.6 port 54545 connected with 10.0.0.18 port 5001
[  9] local 10.0.0.6 port 54546 connected with 10.0.0.18 port 5001
[ 11] local 10.0.0.6 port 54548 connected with 10.0.0.18 port 5001
[ 12] local 10.0.0.6 port 54549 connected with 10.0.0.18 port 5001
[  5] local 10.0.0.6 port 54542 connected with 10.0.0.18 port 5001
[ 10] local 10.0.0.6 port 54547 connected with 10.0.0.18 port 5001
[ 12]  0.0- 2.8 sec    408 KBytes  1.18 Mbits/sec
[ 11]  0.0- 3.4 sec    408 KBytes    990 Kbits/sec
[ 10]  0.0- 4.4 sec    408 KBytes    762 Kbits/sec
[  6]  0.0- 7.2 sec    408 KBytes    465 Kbits/sec
[  5]  0.0- 7.2 sec    408 KBytes    462 Kbits/sec
[  9]  0.0-14.9 sec    408 KBytes    224 Kbits/sec
[  3]  0.0-28.9 sec    408 KBytes    116 Kbits/sec
[  7]  0.0-57.1 sec    408 KBytes  58.5 Kbits/sec
```

Figura 6-25. Segundo test iPerf con un 20% de pérdida de paquetes. (Fuente: elaboración propia).

Por último, se debe tener en consideración que los resultados obtenidos en la prueba de rendimiento, generados por iPerf en VIRL, sólo deben utilizarse para fines de prueba de concepto. Puesto que la red y routers virtuales simulados, no poseen las mismas características de hardware que las redes reales, por lo tanto no reflejan el real comportamiento.

6.3 Conectividad externa

En el siguiente bloque, se verá la estructura de VIRL desde el punto de vista de la conectividad, con el objetivo de comprender el funcionamiento que éste posee para acceder a las simulaciones y nodos. Esta visión permitirá entender cómo funciona dicha conectividad y por tanto las alternativas de conexión con redes reales, permitiendo ampliar las posibilidades de crear topologías más avanzadas.

6.3.1 Estructura de VIRL

El servidor VIRL es un hipervisor habilitado para KVM (*Kernel-based Virtual Machine* o máquina virtual basada en el núcleo), donde todos los nodos simulados corren dentro de éste. Al momento de correr VIRL en VMware Workstation Pro se crean dos niveles bajos de entorno virtual (ver figura 6-26). Esto es denominado Virtualización Anidada. Es por esta razón que el PC donde se correrá VIRL debe tener soporte de tecnología Intel VT-X o AMD-V para procesadores Intel y AMD respectivamente.

KVM, es un módulo del kernel de Linux que permite utilizar características de virtualización de procesadores como Intel y AMD a programas con espacio de usuario.

QEMU es un hipervisor basado en el kernel, el que gestiona los nodos simulados y los recursos de máquinas virtuales. El único problema de QEMU es que el proceso de comunicación entre la CPU invitada y la CPU del host es extremadamente lenta [30].

Kernel Virtual Machine (KVM) es un acelerador que ayuda a QEMU. Con la extensión de virtualización de hardware permite que las máquinas virtuales accedan al hardware de la máquina física directamente con un mínimo de sobrecarga y delay.

La figura 6-26, ilustra la estructura de VIRL corriendo en hipervisor VMware. Donde la parte inferior corresponde al hardware con soporte de tecnología VT-X/AMD-V, pasando por el hipervisor VMware donde se aloja Ubuntu en conjunto con QEMU y KVM. Por último se tienen los nodos simulados como máquinas virtuales basadas en el kernel como se explicó.

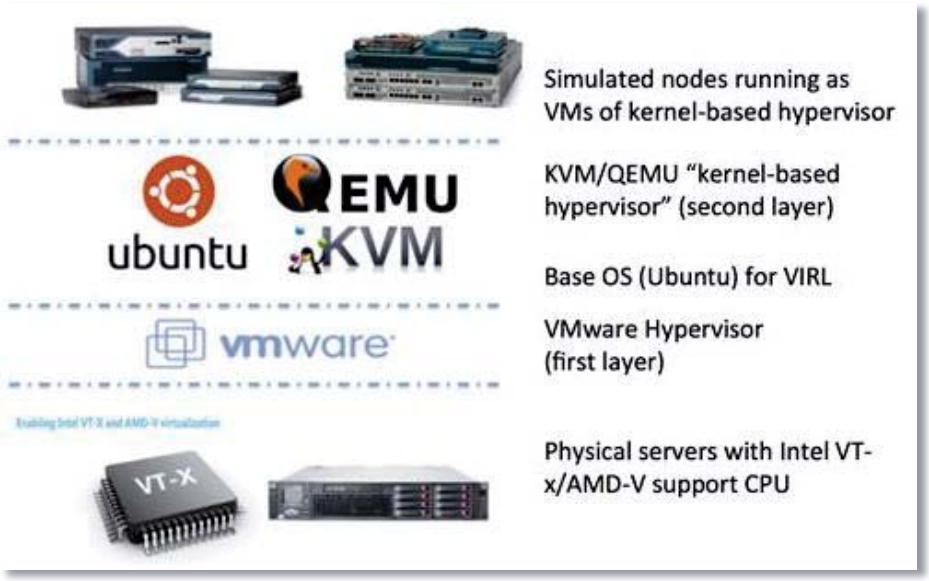


Figura 6-26. Estructura de VIRL desde la capa física con tecnología VT-X/AMD-V hasta los nodos simulados. (Fuente: https://www.speaknetworks.com/)

En 6.1, se vio que VIRL depende de la utilización de RESTful APIs para su funcionamiento. OpenStack es quien crea, elimina y administra máquinas virtuales y sus recursos de acuerdo con las llamadas API y comandos introducidos mediante CLI, tal como se vio.

OpenStack es una plataforma de software de código abierto para *cloud computing*, en su mayoría desplegada como IaaS (*infrastructure-as-a-service*). La plataforma de software consiste en componentes interrelacionados que controlan el conjunto de hardware para el procesamiento, almacenamiento y recursos de red a través de un centro de datos. La figura 6-27, ilustra la distribución de OpenStack donde pueden apreciarse algunos servicios como Neutron, Nova y otros.

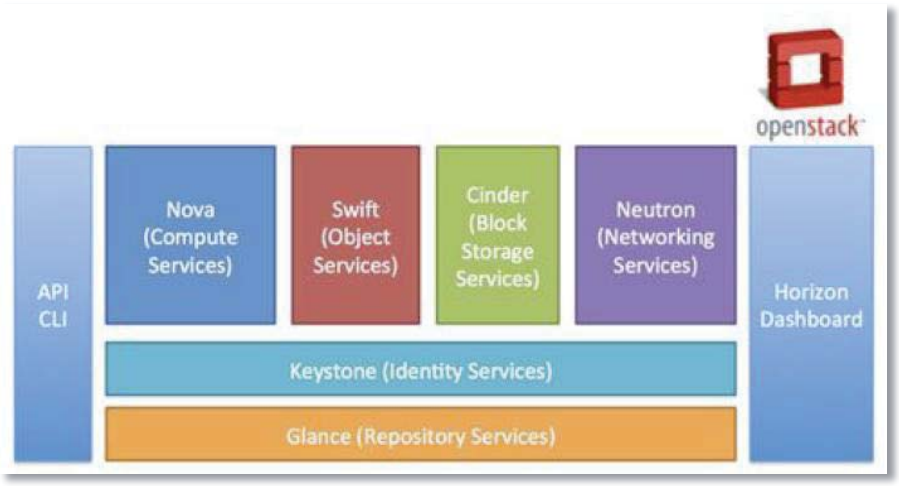


Figura 6-27. Estructura de OpenStack y sus servicios. (Fuente: https://www.speaknetworks.com/)

Ya que las topologías de red simuladas con VIRL en hipervisor WMware, se encuentran corriendo sobre 2 niveles bajos, según como se muestra en la figura 6-26 ¿Cómo podría realizarse la conexión con el mundo exterior? La información que viene a continuación es crucial para entender las interfaces y redes del servidor VIRL.

Durante la instalación del software, fue necesario configurar 4 redes en el hipervisor VMware Workstation. Estas son, VMnet1, VMnet2, VMnet3, VMnet4, las que fueron asignadas a FLAT, FLAT1, SNAT e INT respectivamente. Además, la red VMnet8/NAT, que es utilizada para ingresar y/o gestionar el software, fue configurada por defecto por VMware Workstation. Esta última interfaz es el primer puente entre la red física y las redes virtuales donde vive VIRL. No se debe enrutar ningún tráfico de simulación a través de esta interfaz. A continuación se explicará cómo los otros puertos (FLAT, FLAT1, SNAT) son utilizados para proveer conectividad externa a las redes simuladas.

En la configuración de las redes virtuales que se realizó durante la instalación del software, fueron asignadas las direcciones IP que se muestran en la figura 6-28. A excepción de VMnet8/NAT, las direcciones IP asignadas a las otras redes vienen por defecto en las instrucciones de instalación.

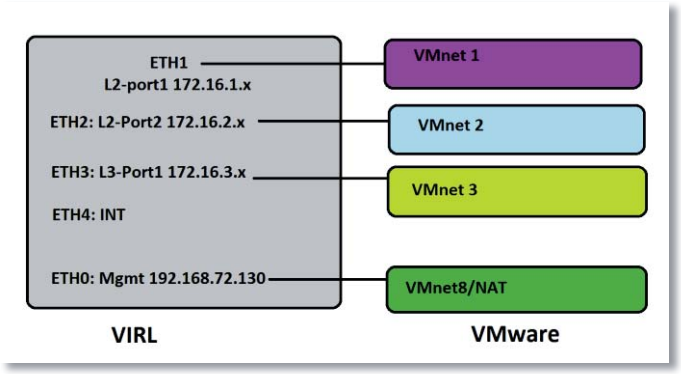


Figura 6-28. Direcciones IP asignadas a las redes virtuales utilizadas por VIRL. (Fuente: elaboración propia).

Anteriormente se vio que cada vez que es utilizado uno de los métodos privados de simulación, es creado un LXC (contenedor Linux). El acceso al manejo de las redes simuladas es posible por el primer acceso que realiza el LXC utilizando protocolo SSH y luego por medio de Telnet o SSH se accede a los nodos (ver figura 6-30). Los LXC también pueden ser configurados para reenviar tráfico a nodos en la simulación o incluso alojar aplicaciones o servicios de red directamente para administrar los nodos [30].

La figura 6-29, ilustra cómo están interconectados VIRL y LXC.

- LXC-ETH0: conectado a VIRL VMnet1 (L2-FLAT) 172.16.1.x/24
- LXC-ETH1: conectado a Management Network 10.255.x.x/16

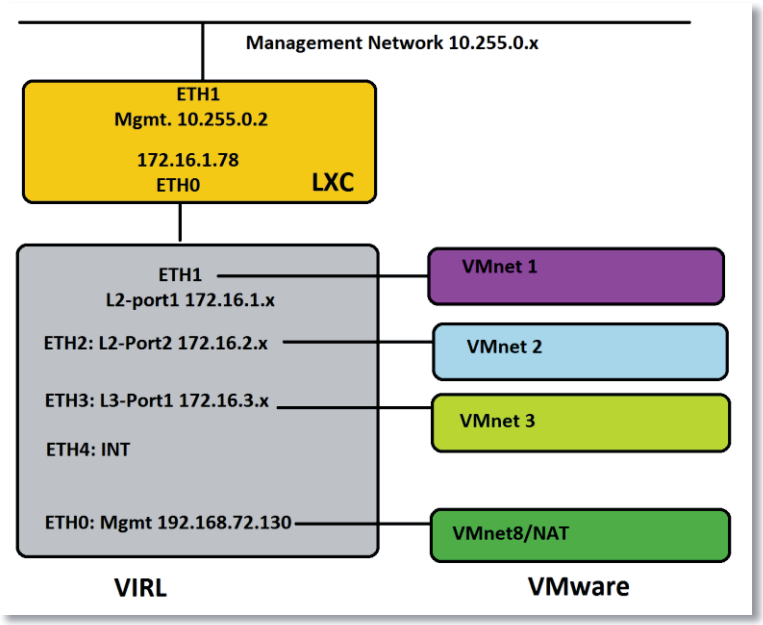


Figura 6-29. Interconexión de LXC y VIRL. (Fuente: elaboración propia).

Cada vez que es utilizado uno de los métodos privados de simulación, todos los servidores y nodos simulados tendrán una dirección IP asignada en la red "Management Network" (identificar en la figura 6-30) donde es conectado el LXC-ETH1. Notar que esta red es diferente a la utilizada para acceder a VIRL (VMnet8 en la figura 6-29).

Entonces VIRL, LXC y la Simulación trabajan juntos de la manera en que se indicó anteriormente. La figura 6-30, representa una simulación donde se aprecian las conexiones con las diferentes interfaces de VIRL. En la parte superior de la figura 6-30, sobre la red "Management Network", se encuentran los nodos simulados: IOSv, ASAv, IOSvL2 y los servidores; bajo esta red, se encuentra VIRL (cuadro celeste), el cual está conectado al hipervisor VMware en sus diferentes interfaces (4 cuadros de colores) y al LXC (cuadro verde) respectivo a una simulación "privada". Las nubes "Snat" y "flat", se conectan a la simulación tal como lo indica la línea segmentada. Además, observar que las interfaces de manejo de cada nodo se conectan con el LXC. Esta interfaz de manejo no participa en el tráfico de data-plane, esta es designada para gestionar el nodo únicamente. Desde la perspectiva del usuario, se puede acceder a los nodos mediante la interfaz de gestión o por medio de la dirección IP del puerto de red de datos. Por ejemplo, podría accederse a realizar configuraciones al nodo ASAv-1 de la figura 6-30, por medio de la interfaz de gestión (10.255.0.3) o una de las interfaces de datos. La misma teoría es aplicada para acceder a dispositivos de redes físicas.

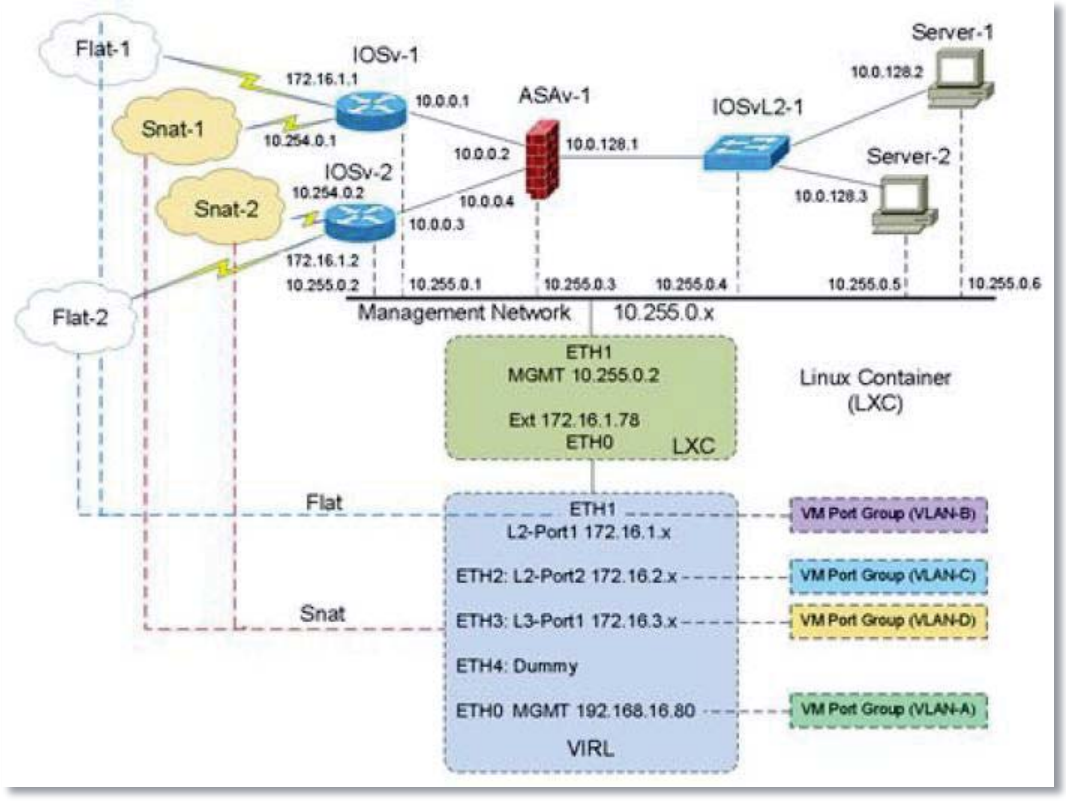


Figura 6-30. Conectividad entre VIRL, LXC y nodos de una simulación "Privada". (Fuente: https://www.speaknetworks.com/)

En 5.4, se vio las diferencias entre cada uno de los métodos de simulación. En resumen se tiene:

- **Private Simulation Network**:
    - Private simulation posee su propio LXC. El LXC sólo tiene conexión con los nodos que están corriendo en la simulación.
    - El LXC no puede ver y por tanto no puede acceder a los nodos en cualquier otra simulación, incluso en simulaciones que forman parte del mismo proyecto.

- **Private Project Network**:
    - Private Project comparte un LXC, incluso para múltiples simulaciones que sean parte del mismo proyecto.
    - El LXC no puede ver y por tanto no puede acceder a nodos en ningún otro proyecto.

Observar que el uso del LXC no es solo como una "caja" conveniente, también es utilizado para crear una barrera que separe múltiples simulaciones o proyectos en un entorno de laboratorio compartido.

- **Shared FLAT Network**:
    - El método de simulación Shared FLAT Network elimina la necesidad de un LXC.
    - Las interfaces de gestión de los nodos en la simulación están conectados directamente en la red FLAT 172.16.1.0/24 (ver figura 5-16).
    - Los nodos son visibles para todos los otros nodos en cualquier simulación, cualquier proyecto y cualquier usuario.
    - VIRL puede tener acceso directo a todos los nodos simulados por medio de la interfaz ETH1 en la red FLAT (ver figura 6-31).

La figura 6-31, ilustra una simulación con el método Shared FLAT Network. En la parte superior, sobre la subred 172.16.1.x (notar que no es la misma que la figura 6-30), se encuentran los nodos simulados: IOSv, ASAv, IOSvL2 y los servidores. VIRL está en el mismo lugar que en la figura 6-30 (cuadro celeste), el cual está conectado al hipervisor VMware en sus diferentes interfaces (4 cuadros de colores), estos enlaces permanecen siempre de la misma manera. Como se está utilizado el método "Shared", no es creado un LXC como en el caso anterior, y las interfaces de gestión de los nodos están conectadas directamente en la interfaz ETH1 de VIRL, por medio de la subred FLAT. Debido a esto, la red FLAT no puede ser utilizada al mismo tiempo para conectividad de data-plane y gestión de nodos. Si se utiliza la subred FLAT para conectividad de data-plane, se debe seleccionar un método privado de simulación para el manejo de los nodos, ya que de esta manera se accedería a las interfaces de gestión por medio de la subred 10.255.0.x, tal como se ilustra en la figura 6-30, sin interferir en la red FLAT.

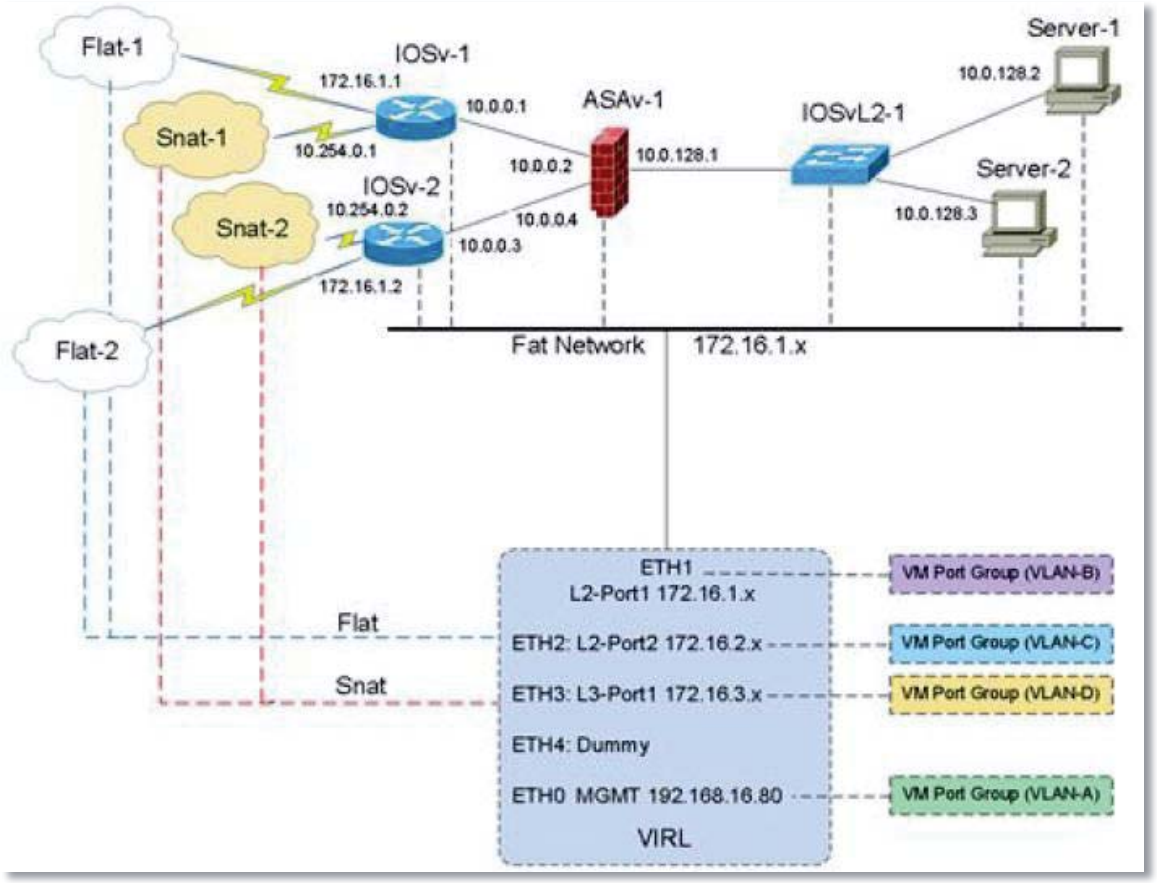


Figura 6-31. Conectividad entre VIRL y nodos de una simulación *"Shared"*. (Fuente: https://www.speaknetworks.com/)

El siguiente ejemplo expone un "caso real" sobre la utilización de los métodos de simulación.

Un profesor de la EIE de la PUCV está dictando una asignatura de redes. En la clase hay 20 estudiantes pero la Universidad sólo aprobó la compra de una única licencia VIRL. Los estudiantes utilizarán VIRL para aprender comandos CLI Cisco y realizar las tareas asignadas.

- El Profesor debe dividir a los 20 estudiantes en 5 grupos de 4 personas.
- Cada grupo es asignado con un proyecto de red, donde los 4 estudiantes colaborarán para elaborar un informe con una solución final.
- Cada estudiante podrá practicar lanzando simulaciones a pequeña escala. Los estudiantes no podrán ver otras simulaciones individuales.

Con estos requerimientos, el Profesor configura VIRL de la siguiente manera:

Cada estudiante tendrá una "simulación privada", donde el LXC también será privado. De esta manera, lo que haga un estudiante no afectará a sus compañeros.

Cada grupo tendrá un “Proyecto Privado”, donde un LXC es compartido entre compañeros del mismo grupo. Los 4 compañeros podrán ver y trabajar en el mismo proyecto. Cada grupo podrá ver únicamente su propio proyecto.

6.3.2 Conectividad con el mundo exterior

VIRL provee métodos para conectar redes simuladas con el mundo exterior. Entiéndase como “mundo exterior” a que VIRL se conecta a un host distinto al que se encuentra instalado, para no confundir con experiencia N° 5, donde se hace referencia al “mundo exterior” aludiendo al uso de la nube “flat-1”, a pesar de que esta herramienta permitirá dicha conexión.

El primer método es sobre la red FLAT. Este crea una red común de capa 2 en la misma subred 172.16.1.x. que cruza al entorno virtual y físico por medio de la interfaz ETH1 del servidor VIRL. El segundo método es por medio de la red SNAT. Este método crea un enlace estático que proporciona NAT entre entornos físicos y virtuales a través de la interfaz ETH2 del servidor VIRL [30].

Las herramientas de conectividad externa (*“External Connection Tool”*) que se aprecian en VM Maestro (ver figura 6-32), son utilizadas para crear una o más conexiones de capa 2 (FLAT) o capa 3 (SNAT) desde la red simulada hacia el mundo exterior por medio de la interfaz Ethernet de VIRL.

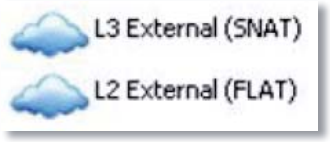


Figura 6-32. Herramientas de conectividad externa. (Fuente: https://www.speaknetworks.com/).

Considérese la topología de la figura 6-33. La simulación de esta topología se configuró para utilizar el método de simulación “privada”, lo que implica que se crea el respectivo LXC. El foco será sólo sobre el nodo iosv-1 y su conectividad con el mundo exterior.

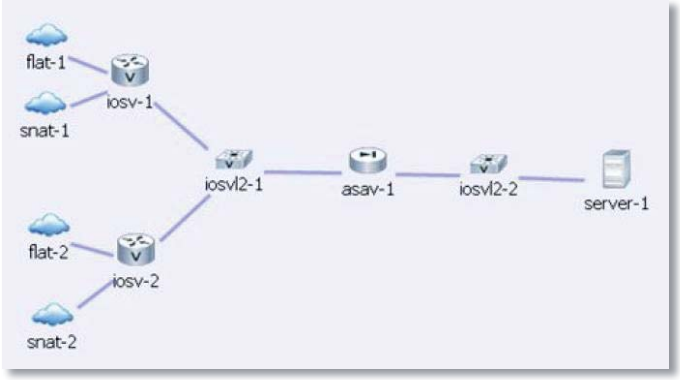


Figura 6-33. Topología de ejemplo. (Fuente: https://www.speaknetworks.com/).

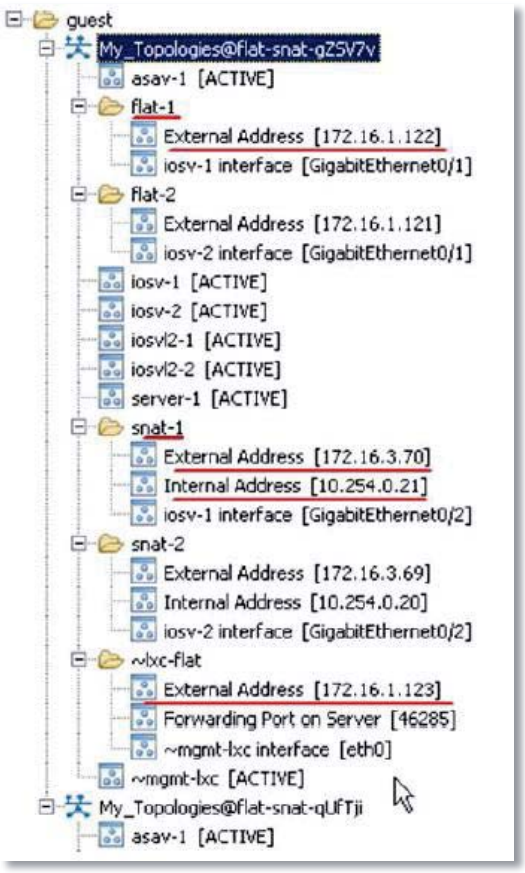


Figura 6-34. Direcciones IP asignadas a la simulación de la topología mostrada en la figura 6-33. (Fuente: https://www.speaknetworks.com/).

Las direcciones IP importantes para el nodo iosv-1 son:

- Gig0/1: 172.16.1.122 conectado a FLAT-1 exterior.
- Gig0/2: 10.254.0.21 conectado a SNAT-1 red exterior.
- Gig0/0: 10.255.0.144 conectado a la red de gestión LXC (esta interfaz no participa en el tráfico de datos).

El nodo iosv-1, posee dos maneras de conectarse con el mundo exterior. La primera es mediante la interfaz Gig0/1 172.16.1.22 sobre la red “Flat1”. La segunda es mediante la interfaz Gig0/2 10.254.0.21 sobre la red “SNAT”.

Conexión de iosv-1 por medio de la red FLAT-1

La interfaz Gig0/1 se encuentra conectada directamente a la red Flat1, donde VIRL también posee una interfaz. Para realizar pruebas, debe configurarse un gateway por defecto en iosv-1, para dirigir todo el tráfico saliente hacia la red FLAT-1.

Luego de esto, es posible realizar un ping a 172.16.1.254. La nuve "FLAT-1" de la figura 6-33, es básicamente un conmutador de capa 2 de acceso múltiple, el que no participa en ningún tipo de enrutamiento IP.

Conexión de iosv-1 por medio de la red SNAT-1

En este caso, la nuve "SNAT-1" de la figura 6-33, actúa como una máquina de "Nateado" estático. No es posible ver lo que hay dentro de esta nuve, pero básicamente realiza dos cosas:

- Traducción estática de una IP a otra ("Nateado estático").
- Coloca la IP "Nateada" en la interfaz SNAT 172.16.3.x.

Entonces, la nube "SNAT-1" traduce la dirección IP 10.254.0.21 de la interfaz Gig0/2 del nodo iosv-1, a la IP 172.16.3.70 (ver figura 6-34 y 6-35). Además coloca el tráfico traducido en el mismo medio de acceso múltiple que la interfaz 172.16.3.x de VIRL. Tan pronto como esto ocurre, el nodo iosv-1 puede comunicarse con cualquier host en 172.16.3.x.

Hay dos cosas que deben realizarse:

- Configurar una gateway por defecto en iosv-1 y dirigir todo el tráfico a la nuve SNAT-1.
- Configurar una ruta estática en VIRL y dirigir cualquier tráfico de retorno a que vuelva al nodo iosv-1.

La figura 6-35, ilustra los dos métodos que el nodo iosv-1 utiliza para tener acceso a redes externas.

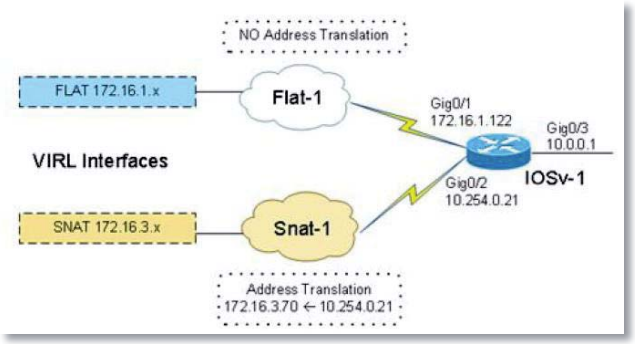


Figura 6-35. Dos métodos de conexión con redes externas del nodo iosv-1. (Fuente: https://www.speaknetworks.com/).

Se dieron los tips para hacer que una red simulada hable con VIRL. Esto quiere decir que la simulación todavía está funcionando dentro de una red autónoma. Entonces ¿Cómo se puede configurar VIRL para que hable con dispositivos de una red externa como un enrutador? La respuesta es que se puede conectar el laboratorio virtual con la infraestructura de red física utilizando la tecnología vSwitch de VMware. El hipervisor VMware Workstation Pro viene provisto con la tecnología vSwitch. Fue utilizada cuando se realizaron las configuraciones de las redes virtuales durante la instalación del software.

Entonces, para realizar dicha conexión, se necesita editar la subred Flat1 para que coincida con el entorno específico de cada usuario. Observar que se está considerando realizar la conexión mediante la red FLAT. Sin embargo, SNAT funciona de manera similar.

Por ejemplo, suponiendo que se tiene un laboratorio con una red física cuya dirección IP es 192.168.17.0/24. Lo que se debe hacer, es cambiar la dirección IP por defecto (172.16.1.x) de la interfaz ETH1 del servidor VIRL a 192.168.17.x (ver figura 7-36). Para esto, se debe configurar la red virtual VMnet1, la cual fue configurada durante la instalación del software (ver figura 3-1) en el "editor de redes virtuales" de VMware Workstation, asignándole la dirección IP correspondiente. En el ejemplo, esta dirección 192.168.17.0/24. La figura 6-36, ilustra cómo quedarían las nuevas direcciones IP, tanto para la interfaz ETH1 como para los nodos IOSv-1 e IOSv-2 luego de realizar los cambios (comparar con la figura 6-30).

Pueden seguirse los mismos pasos para configurar SNAT y conectarse al laboratorio externo. Debe tenerse en cuenta que no se puede utilizar la misma subred para las redes FLAT y SNAT. Según la referencia [30], se añade complejidad con poco o ningún beneficio mediante el uso de SNAT para conectarse a un laboratorio externo. A menos que se desee probar características específicas, se recomienda que se utilice la red FLAT.

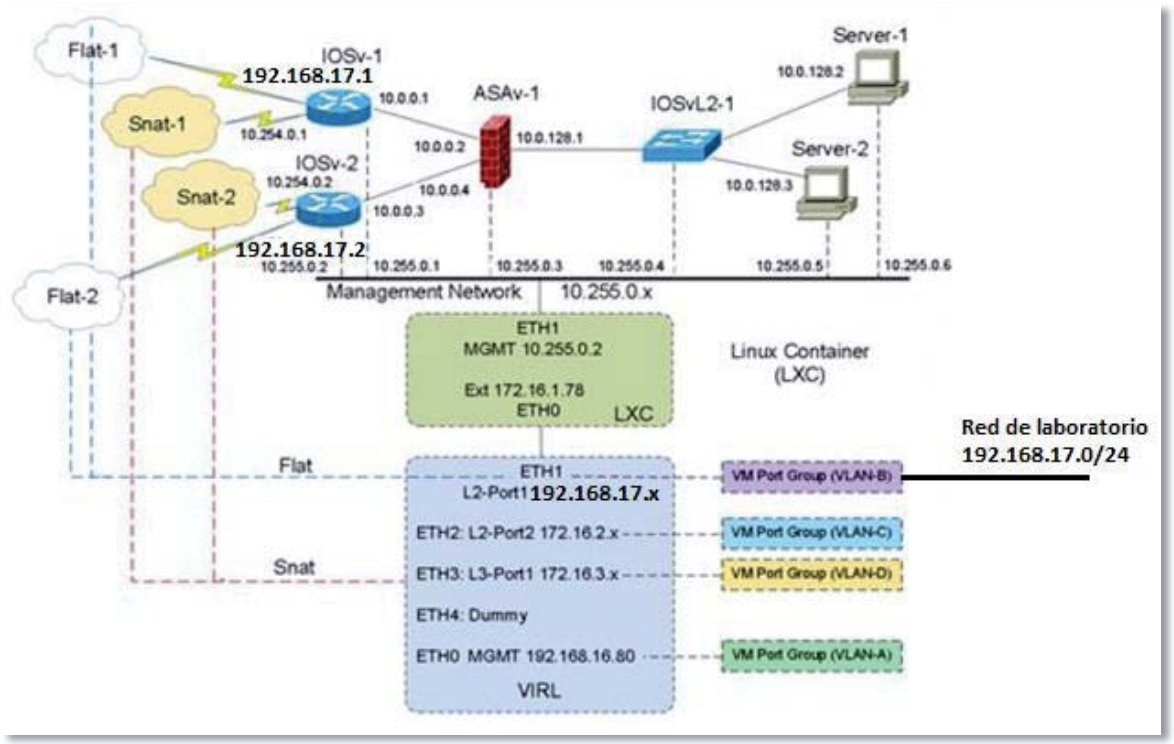


Figura 6-36. Nuevas direcciones IP de la figura 6-30, luego de realizar los cambios para obtener conexión con un entorno de laboratorio real. (Fuente: Edición propia de la figura 6-30).

7 Análisis técnico y ecnómico

Para concluir con el estudio, se realiza un análisis de las características habilitadas en las variaciones de las imágenes cisco IOSv, CSR1000v, IOS-XRv, NX-OSv y ASAv, para conocer las diferencias que habría con equipos reales, además de la factibilidad en correr máquinas virtuales de otros propietarios. Por último, se realiza un estudio económico para la implementación de un laboratorio de redes con VIRL, el cual poseerá capacidad de trabajo para 20 estudiantes simultáneamente v/s un laboratorio con equipamiento real (routers y switches).

7.1 IOS, IOS-XE, IOS-XR y NX-OS en VIRL

Las variaciones de las imágenes Cisco están presentes en VIRL. Estas son: IOSv la que corresponde con IOS, CSR1000v la que corresponde con IOS-XE, IOS-XRv la que corresponde con IOS-XR, NX-OSv la que corresponde con NX-OS. En este bloque, se verán las características que poseen cada una de estas variaciones. Además, se agrega una perspectiva desde las certificaciones Cisco CCNA, CCNP y CCIE, a propósito del vínculo con ambientes de laboratorio.

VIRL, junto con otras herramientas de simulación, no emula ASICs (circuitos integrados de aplicaciones específicas) y dispositivos programable como FPGA. IOSv posee el mismo código de IOS, el cual fue compilado para funcionar en un hipervisor y hacer uso de la virtualización [4]. La única diferencia con un Router real, es que este posee paquetes para ASICs, mientras que IOSv carece de estos paquetes y utiliza software para correrlo en un hipervisor. Es por esta razón que no es posible tener algunas de las características que pueden encontrarse en hardware real, puesto que ciertas funciones son difíciles de emular en software. Sin embargo, configuraciones y funcionalidades del software original IOS son idénticas en IOSv.

7.1.1 Características de IOSv, CSR1000v, IOS-XRv, NX-OSv y ASAv

A continuación se presentan las características soportadas para IOSv, IOSvL2, CSR1000v, IOS-XRv, NX-OSv y ASAv [4].

- **IOSv (IOS):** 802.1Q, AAA, ACL, BGP, DHCP, DNS, EEM, EIGRP, EoMPLS, Flex Netflow + TNF, GRE, ICMP, IGMP, IP SLA, IPSec, IPv6, ISIS, L2TPv3, MPLS, MPLS L2VPN, MPLS L3VPN, MPLS TE, Multicast, NAT, NTP, OSPF, PfR, PIM, PPPoE, RADIUS, RIP, SNMP, SSH, SYSLOG, TACACS, TFTP, VRF-LITE.

- **IOSvL2 (IOS):** Layer-2 forwarding (auto configurado), Switchport (auto configurado), 802.1q trunk, 802.1q VLANs (auto configurado), Spanning Tree (auto configurado), Port-Channel (PAGP y LACP), 802.1x pass-through, Port-ACLs, Dynamic Arp Inspection, DHCP Snooping, IP device tracking, Switched Virtual Interfaces, Layer-3 forwarding sobre SVIs, Routing protocol support, VTP v1-3, PVST, QoS, Inter-VLAN routing, VLAN Access Maps (VACLs / access control lists for VLANs), paquetes de protocolos con funcionalidad ACL para layer2 y layer3, soporte Dynamic Trunking Protocol, modo protegido Switchport.

- **IOS-XRv (IOS-XR):** IPv4, IPv6, BGP, MP-BGP, EIGRP, ICMP, OSPF, NTP, TFTP, MPLS, MPLS L3VPN, MPLS TE, ISIS, mVPN GRE / mLDP / P2MP TE, AAA, RADIUS, TACACS, SNMP, FLEX CLI, Multicast (PIM, MSDP, IPv6), Syslog, VLANs / QinQ (.1Q, .1AD), RPL, ACLs, SSH, VRF-LITE.

- **NX-OSv (NX-OS):** 802.1x, AAA, AMT, BGP, CDP/LLDP, EIGRP, FHRP-HSRP, GLBP, VRRP, ICMP, IGMP, IPv4, IPv4/6, IPv6, ISIS, L3 Routing Protocols, LDAP, LISP, MLD, MSDP, NTP, OSPF, PIM/PIM6, Radius, RIP, SNMP, Syslog, TACACS+, VRF, XML/Netconf, NX-API.

- **CSR1000v (IOS-XE):** 802.1Q, AAA, ACL, BGP, DHCP, DNS, EEM, EIGRP, EoMPLS, Flex Netflow + TNF, GRE, ICMP, IGMP, IP SLA, IPSec, IPv6, ISIS, L2TPv3, MPLS, MPLS L2VPN, MPLS L3VPN, MPLS TE, Multicast, NAT, NTP, OSPF, PfR, PIM, PPPoE, RADIUS, RIP, SNMP, SSH, SYSLOG, TACACS, TFTP, VRF-LITE.

- **ASAv:** soportado sólo en modo de contexto único, no posee soporte para modo de contexto múltiple.
  Características no soportadas: Clustering, Modo de contexto múltiple, Active/Active failover, Ether-Channels, Shared AnyConnect Premium Licenses.

7.1.2 CCNA, CCNP y CCIE con VIRL

Con el ánimo de contrastar la información expuesta en 8.1, y de obtener una visión distinta a sólo poseer una lista con las características disponibles, se expondrá una perspectiva desde las certificantes Cisco CCNA, CCNP y CCIE R&S, ya que las certificaciones Cisco son universalmente reconocidas como un estándar de la industria para diseño y soporte de redes, garantizando altos niveles de conocimientos y credibilidad [31]. Por tanto, revisar las facultades que ofrece VIRL respecto de estas certificaciones, expondrá una visión general del software, formando una idea de las capacidades, sin la necesidad de conocer cada una de las características expuestas en 8.1.

Como ya se explicó, algunas características disponibles en el hardware real no se encuentran en VIRL, simplemente por incompatibilidad de hardware.

Routing y Switching con VIRL

La mayoría de las tecnologías de Capa 3 y superiores requeridas en CCIE R&S son compatibles con IOSv. Algunas características de la capa 2 están ausentes.

Plataformas emuladas:

- IOSv
- Catalyst IOS, Layer 2 IOSv
- CSR1000v y ASR1000, corriendo código IOS XE.

Características:

- **Para capa 2:** VLANs, Trunking, Port-Channels (PAGP y LACP), STP, SVIs, 802.1x pass-through, Port-ACLs, Dynamic Arp Inspection, DHCP Snooping, IP device tracking, VTP v1-3, PVST, QoS, Inter-VLAN routing, VLAN Access Maps (VACLs/access control lists for VLANs), funcionalidad ACL layer 2 y layer 3, soporte para Dynamic Trunking Protocol, modo protegido Switchport.
- **Para capa 3:** RIP, OSPF, EIGRP, ISIS, BGP, PIM, VRF, MPLS L2 and L3, IPSec, DMVPN, 802.1Q, AAA, ACL, DHCP, DNS, EEM, Flex Netflow + TNF, GRE, ICMP, IGMP, IP SLA, IPv6, ISIS, L2TPv3, Multicast, NAT, NTP, PfR, PIM, PPPoE, RADIUS, SNMP, SSH, SYSLOG, TACACS and TFTP.

Como puede verse, es posible virtualizar la mayoría de las funciones principales, logrando reemplazar routers físicos. En términos de características de enrutamiento, casi todo el enrutamiento en IPv4 e IPv6 es compatible con plataformas IOS e IOS-XE.

Proveedor de servicios con VIRL

Plataformas emuladas:

- CRS1000v
- IOS XRv

Características:

- IPv4, IPv6, RIP, OSPF, EIGRP, ISIS, BGP, VRF, VRF-LITE, PIM.
- MPLS L2VPN, MPLS L3VPN, MPLS TE, ISIS, mVPN GRE / mLDP / P2MP TE.
- Multicast (PIM, MSDP, IPv6), Syslog, VLANs / QinQ, MPLS Traffic Engineering.

Con IOS XRv ejecutándose en VIRL, se admite un 90% o más de las funciones de enrutamiento.

Seguridad con VIRL

Plataformas emuladas:

- ASAv

- IOSv, CSR1000v
- Layer 2 IOSv

Características:

- IOS ACLs, NAT, ZBPF, IPSec, EzVPN, DMVPN, ASDM.
- ASA MPF (Modular Policy Framework), NAT, L2L IPsec VPN, RA IPsec VPN.
- Cisco Secure ACS, Identity Services Engine (ISE), vWSA, vWLC puede ser importado como una VM invitada corriendo in VIRL.

En ASAv, no están soportados: Clustering, modo de contexto múltiple, Active/Active failover, EtherChannels, Shared AnyConnect Premium Licenses.

Centro de datos con VIRL

Plataformas simuladas:

- NX-OSv
- IOSv, CSR1000v
- Layer 2 IOSv

Características:

- RIP, EIGRP, OSPF, ISIS, BGP, PIM VRFs.
- 802.1x, AAA, AMT, CDP/LLDP, FHRP-HSRP, GLBP, VRRP, ICMP, IGMP, IPv4, IPv4/6, IPv6, LDAP, LISP, MLD, MSDP, NTP, PIM/PIM6, Radius, SNMP, Syslog, TACACS+, VRF, XML/Netconf, NX-API.

Observar que un mínimo de características de capa 2 están soportadas por la plataforma Nexus NX-OSv. Por ejemplo, algunas características de capa 2 implementadas en hardware ASICs como vPort-channel, OTC, Fabric path, no están soportadas.

Para tener una idea más clara, el software para simular redes de datos, Cisco Packet Tracer, apoya la mayoría de los protocolos y tecnologías enseñados en los programas de Cisco (CCNA Discovery, CCNA Exploration y CCNA Security), y puede ser usado para enseñar conceptos esenciales de TI y Cisco CCNP, no incluyendo CCIE [32]. En contraste con VIRL, este posee muchas de las características que se encuentra en el hardware real, excluyendo sólo algunas funcionalidades, permitiendo incluso ocupar VIRL para preparar CCIE R&S.

7.2 ¿Es posible integrar máquinas virtuales de terceros a VIRL?

Luego de revisar la documentación disponible en [33], la respuesta simple a si es posible integrar VMs de terceros, es Sí. Dentro de los documentos de apoyo a la comunidad VIRL, se encuentra una lista con VMs de terceros que han podido ser incluidas en el software.

Será desplegada una lista con las VMs de terceros que han podido ser integrar. Implementar máquinas virtuales de terceros, consta de una serie de pasos técnicos. Sin embargo, se encuentra información disponible de cómo llevar a cabo esta implementación.

La lista siguiente corresponde a las VMs de terceros que han podido ser integradas a Cisco VIRL por parte de miembros de la comunidad VIRL.

- Arista vEOS
- Alcatel 7750 SR
- Citrix Netscaler
- Cumulus VX
- Extreme Networks
- F5 BIG IP
- Forescount CounterACT
- Fortinet FortiGate (firewall)
- HP VSR1k - HP VSR1000
- Juniper vMX
- Juniper vSRX
- Kali Linux
- Palo Alto Networks
- Security Onion
- Windows - Windows 7 (Host)

La lista anterior, que se encuentra en [33], está en constante actualización, fue modificada por última vez el 02 de junio de 2016.

7.3 Análisis económico para la implementación de un laboratorio de redes con Cisco VIRL v/s equipamiento real

Luego de haber conocido prácticamente la totalidad de las competencias del software, y considerando que la mayoría de las características están disponibles en las imágenes Cisco, tal como se vio en 7.1, donde sólo están ausentes algunas características de capa 2, principalmente en IOSv, como se indica en 7.1.2. Por lo que se plantea realizar un estudio económico de costo anual uniforme equivalente (CAUE), para la implementación de un laboratorio de redes con capacidad para 20 estudiantes trabajando simultáneamente, comparando el CAUE de la implementación con Cisco VIRL v/s equipamiento real.

7.3.1 Antecedentes

Solución con Cisco VIRL

Según el acuerdo de licencia de Cisco VIRL:

"Una licencia está asociada a un único usuario. Se permite que cada licencia tenga hasta dos instalaciones, ya que muchos usuarios han solicitado la capacidad de tener 2 instalaciones, una

en el trabajo o un servidor y otra en su computadora portátil para la portabilidad. Sin embargo, todavía se solicita que no se ejecuten ambas instalaciones simultáneamente; quienes realicen esto, serán capturados por el sistema, tratándose de una violación al acuerdo de licencia." [30]

Esto quiere decir que no es posible tener el software instalado y corriendo en varios PC con una misma licencia. Sin embargo, se permite ejecutar dos instalaciones, siempre y cuando no estén al mismo tiempo.

Se podría adquirir 20 licencias para poder implementar un laboratorio con capacidad de trabajo para 20 estudiantes simultáneamente. Sin embargo, VIRL es una aplicación cliente-servidor, esto quiere decir que VIRL puede ejecutarse en un único servidor (PC) y acceder a él desde varios equipos, como clientes VM Maestro.

En las actividades realizadas en capítulos anteriores, se accedía al servidor VIRL desde el PC host, como cliente VM Maestro por medio de la utilización de RESTful APIs. En este caso, se tienen 20 clientes VM Maestro, los cuales se encuentran instalados cada uno en un PC. Estos 20 clientes VM Maestro, accederán al servidor VIRL mediante una red LAN con topología estrella, como la ilustrada en la figura 7-1. La configuración de la red para tener acceso y manejo sobre VIRL, será de manera similar a como se realiza la conectividad exterior en 6.3, con la diferencia que ahora se debe realizar las configuraciones para enviar el tráfico por medio de VLAN-A de la figura 6-36, ya que es por aquella interfaz virtual donde se realiza el manejo del software, por medio de la utilización de RESTful APIs, tal como se vio.

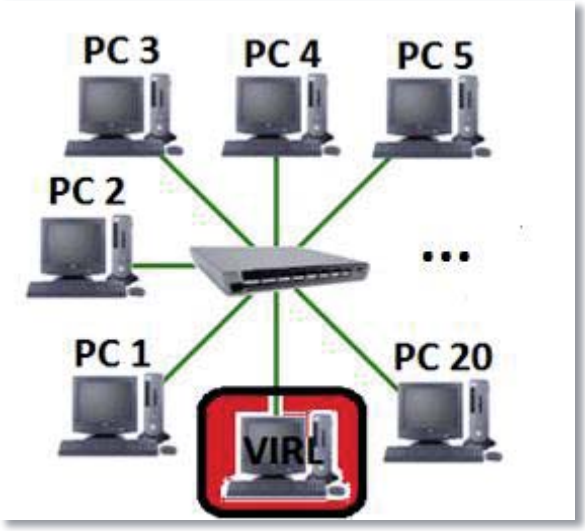


Figura 7-1. Topología en estrella. Solución Cisco VIRL. (Fuente: elaboración propia).

La figura 7-1, ilustra la topología que posee la implementación de un laboratorio de redes con Cisco VIRL, para 20 usuarios trabajando simultáneamente.

Por lo que será necesario considerar (detalles en tabla A-1 del apéndice A):

- PC donde estará alojado VIRL

- 20 estaciones de trabajo (clientes VM Maestro): 20 PC básicos con sus respectivos escritorios y sillas, además de los periféricos
- 1 Switch D-Link DES-1024D (24 puertos RJ45, son necesarios 21), para la interconexión entre las estaciones de trabajo y el servidor VIRL
- Cableado estructurado

De la tabla A-1 del apéndice A, se obtiene que la inversión inicial es de $11.838.494, IVA incluido, más la licencia del software, la cual posee un costo de $199.9 USD por año [4] ($134.245 CLP aprox.). Se considera un valor residual de los equipos del 15% de la inversión inicial, puesto que la depreciación de estas tecnologías es rápida, quedando un valor de $871.716.

Solución con equipamiento real

Ya que VIRL posee la capacidad de simular 20 nodos simultáneamente, la cantidad de dispositivos a utilizar no debe ser superior a 20, para poder comparar ambas soluciones.

Por lo que se considerará (detalles en tabla A-2 del apéndice A):

- 6 Router CISCO1921-SEC/K9
- 3 Router CISCO2901-SEC/K9
- 2 Switch WS-C2960-24TC-S
- 1 Switch Cisco WS-C2960S-24TS-S
- 1 ASA5508-K9
- 5 PC básicos para conectarse a los equipos junto con sus escritorios, sillas y periféricos
- Un rack para alojar los módulos
- Cableado estructurado

De la tabla A-2 del apéndice A, se obtiene que la inversión inicial es de $24.689.850, IVA incluido. Se considera un valor residual de los equipos del 20%, lo que corresponde con el precio en el mercado local de artículos usados, quedando un valor residual de $3.191.965.

Cálculos

Se considera una tasa de retorno mínima aceptada (TRMA) del 8%, y un plazo de 5 años para realizar la proyección.

CAUE$_{VIRL}$ = 9.835.504*(A/P, 8%, 5) - 871.716*(A/F, 8%, 5) + 134.245 = 11.704.249*0,25046 – 871.716*0,17046 + 134.245 = $2.449.053.

$2.449.053 de costo anual uniforme equivalente.

CAUE$_{equipo}$ = 20.747.773*(A/P, 8%, 5) – 3.191.965*(A/F, 8%, 5) = $4.652.385

$4.652.385 de costo anual uniforme equivalente.

Conclusión estudio económico

Se tiene que el $CAUE_{VIRL} < CAUE_{equipo}$. Por lo tanto, es más conveniente implementar el laboratorio mediante el software que implementarlo con equipamiento real, considerando el proyecto durante 5 periodos de un año y una TRMA del 8%.

Discusión y conclusiones

Comenzando con el estado del arte, fue posible corroborar la hipótesis inicial de que Cisco VIRL, presenta las mejores características respecto de software para simular redes de datos, según la Tabla 1-1. Además, Queda claro que la virtualización de recursos de hardware, es el motor fundamental, que le permite al software lograr altos desempeños en las simulaciones. Es por esto que es necesario la utilización de procesadores x86, que posean tecnología de virtualización. Puesto que sin esta tecnología, no sería posible lograr que VIRL funcione, debido a su naturaleza de hipervisor. La herramienta VMware está sobre el sistema operativo y es indispensable para correr VIRL.

Realizar cambios, actualizaciones, nuevas configuraciones a redes que se encuentran en funcionamiento, representa un reto para los ingenieros ya que interrumpir el funcionamiento de estas no es una opción, o simplemente las nuevas características o servicios a implementar podrían traer consecuencias indeseadas o imprevistas. La virtualización de redes permite tener una representación fiel de una red real. Esto es una gran ventaja, ya que no se necesita de hardware adicional para realizar pruebas sobre las nuevas configuraciones, no hay que cablear, la energía eléctrica requerida no se compara con la implementación de hardware, no se necesita refrigeración, personal adicional, etc. Es posible tener un laboratorio multimarca en VIRL. De hecho, cualquier máquina que sea compatible con un hipervisor soportado, puede ser integrada a VIRL. Todo lo anterior se traduce en ahorrar dinero y tiempo.

Una de las desventajas de la virtualización es que se necesitan bastantes recursos de hardware, CPUs potentes y gran capacidad de memoria. Esto resulta lógico si se piensa que se están ejecutando varios sistemas operativos (máquinas virtuales) en un mismo PC.

Una red de datos está constituida por varios dispositivos (routers, switches, etc.) que están interconectados. Esto quiere decir que a la hora de realizar la virtualización, se deben crear comandos con todas estas características de la red para la orquestación de las máquinas virtuales y sus respectivas conexiones. VIRL posee 3 motores que permiten realizar estas tareas. Un motor de simulación basado en Openstack, un motor de configuración basado en AutoNetkit y la interfaz de usuario que brinda el entorno de simulación. Las partes también pueden ser manejadas por terceros mediante archivos XML o librerías de funciones a las que se accede por protocolo HTTP.

Adaptar las nuevas configuraciones o servicios a una red existente se torna simple con el uso de imágenes idénticas a las utilizadas en máquinas físicas. VIRL permite realizar cambios y registrar estos a medida que se van concretando. Es decir que, si se tiene una simulación de una red, y se quiere implementar cambios, se pueden realizar paso a paso e ir registrándolos.

Contar con un laboratorio virtual presenta muchas ventajas. Comenzando con que no es necesario tener hardware costoso para realizar pruebas, el tiempo requerido para montar una red virtual es mucho menor que hacerlo de manera física, sobre todo con la interfaz gráfica que proporciona VIRL, donde es necesario sólo el mouse para colocar dispositivos. Otra característica que se obtiene con un laboratorio virtual, es que se puede ver qué está sucediendo en las diferentes capas, a diferencia del mundo real donde esto no es posible. Otra facultad de VIRL es que una configuración de una red compleja puede ser guardada en un archivo. Es decir que podría llevarse toda la configuración de una red extensa en un pendrive, hacer copias, etc. Otra característica importante implícita, es que se cuenta con las mismas características de data-plane, control-plane y management-plane, es decir que es como si se estuviese frente al dispositivo físico.

VIRL es soportado sólo por hipervisores otorgados por VMware Inc. Además, como VIRL proporciona la capacidad de vincular las interfaces tanto de management-plane y data-plane para nodos dentro de la simulación y dispositivos de redes externas, es que requiere una interfaz compartida entre el PC host y la máquina virtual VIRL, y sólo VMware Workstation Pro proporciona esta herramienta cuando se trabaja con un sistema operativo anfitrión.

VIRL requiere conexión con servidores Cisco SaltStack por lo menos una vez por semana, es decir que debe conectarse a internet cada 7 días por lo menos, puesto que de otra manera, VIRL no permitirá realizar simulaciones fuera del tiempo señalado. Fue necesario que el administrador de red, habilitara los puertos TCP 4505 y 4506 para poder correr simulaciones por el motivo que se indicó.

En la sección 4.2, se pudo observar que VIRL posee una gran flexibilidad de direccionamiento. En la figura 4-3 se aprecia esta flexibilidad. Esto puede ser de utilidad para realizar configuraciones manuales o probar configuraciones, además de la integración de redes existentes.

En la sección 4.5, se concluye que poner en marcha una simulación podría tardar varios minutos, esto es porque cada nodo debe pasar a un estado activo. Luego de que se encuentran activos los nodos, es posible acceder a la consola. Mientras no termine la inicialización de la simulación, no será posible acceder a consolas, incluso si el nodo de la consola a la que se quiere acceder se encuentra en estado activo. Cuidar que el procesador no se encuentre trabajando al 100% una vez iniciada la simulación, puesto que de ser así, podría no traspasarse de manera correcta la configuración de algún nodo desde el archivo de topología. Se recomienda inicializar un máximo de 5 nodos IOSv simultáneamente y los restantes ponerlos en marcha una vez que se haya iniciado la simulación. Esto evitará que el procesador trabaje al 100%, cuidando que no puedan ocurrir errores en el traspaso de configuraciones a las VMs.

En la sección 5.1, los cambios realizados a mano o extraídos de una simulación, se guardan y se utilizan en la próxima vez que se inicie la simulación. Se debe tener en cuenta que al realizar las configuraciones automáticas, el archivo de configuración se sobre escribirá. Para evitar perder configuraciones realizadas a mano o extraídas de una simulación, poner cuidado en la activación o desactivación en la generación de configuración automática para los nodos que se desee.

En la sección 5.2, fueron configurados nueve routers con 3 IGPs e implementados route-reflectors BGP en tan solo unos minutos. Esta misma tarea podría haber tomado fácilmente medio día con hardware real.

En el bloque 6.1, se vio que VIRL depende principalmente para su funcionamiento de RESTful APIs. Estas son aplicaciones que consisten en una arquitectura de software-comunicación que define la manera de cómo operará. La utilización de este tipo de aplicaciones permite mejorar la escalabilidad, el rendimiento y la portabilidad. La mayoría de los servicios disponibles en VIRL están basadas en este tipo de aplicaciones. Donde se ejecutan llamadas a estas y se utilizan sus respuestas. En UWM (User Workspace Management), se encuentra toda la información referente a las APIs VIRL. Es posible encontrar una descripción de la función que realiza cada API, los verbos o métodos que utilizan cada API, la clase de respuesta, entre otros. Esta información facilita la utilización de las RESTful APIs en VIRL. Además se vio que la herramienta Python es la más idónea para realizar automatizaciones en las simulaciones debido a la disponibilidad de bibliotecas que permiten un alto manejo en servicios basados en la web como las RESTful APIs.

En la sección 6.3, se vio que es posible conectar VIRL con redes y dispositivos del mundo exterior. Esto es posible gracias a la utilización de las redes virtuales que el hipervisor VMware Workstation Pro le otorga al servidor VIRL. Estas redes son utilizadas para el manejo de los nodos en la simulación, conexión de capa 2 (Flat y Flat1) y capa 3 (SNAT).

En la sección 6.2, se vio que es posible inyectar perdida de tráfico, latencia y jitter en una simulación. Además de realizar captura de paquetes en vivo y su posterior disponibilidad para análisis. Estas características le brindan a VIRL facultades que ayudan a resolver problemas o la visualización de posibles escenarios. Además, VIRL provee la herramienta iPerf, quien permite medir ancho de banda o rendimientos de una red simulada. Pero se debe tener en consideración que los resultados obtenidos en la prueba de rendimiento, generados por iPerf, sólo deben utilizarse para fines de prueba de concepto. Puesto que la red y routers virtuales simulados, no poseen las mismas características de hardware que las redes reales y por lo tanto no reflejan el real comportamiento.

En el capítulo 7, se vio que VIRL, junto con otras herramientas de simulación, no emula ASICs (circuitos integrados de aplicaciones específicas) y dispositivos programable como FPGA. Por tanto no están disponibles algunas de las funcionalidades de IOS en IOSv, al igual que las demás variaciones de las imágenes Cisco. Sin embargo, estas funciones ausentes son mínimas. Esto queda demostrado cuando se analizan las posibilidades para preparar las certificaciones CCNA, CCNP y CCIE R&S con VIRL, lo que es un indicador claro de las capacidades del software, a diferencia del software Cisco Packet Tracer quién sólo permite preparar CCNA Discovery, CCNA Exploration y CCNA Security), además de ser usado para enseñar conceptos esenciales de TI y

Cisco CCNP, no incluyendo CCIE. Se concluye que, en general, algunas características de capa 2 no están disponibles para los nodos VIRL.

Finalmente, luego de conocer prácticamente la totalidad de las características del software, se concluye que cuenta con las herramientas y recursos suficientes para satisfacer las necesidades planteadas en 1.1. Convirtiéndose en la herramienta más potente para simular redes de datos.

Implementar un laboratorio de redes con capacidad para 20 estudiantes trabajando simultáneamente con Cisco VIRL, es factible debido a su naturaleza. Según el estudio económico realizado en 7.3, es más conveniente implementar un laboratorio con esas características, utilizando el software en lugar de equipamiento real.

Un laboratorio de redes con VIRL, trabajando con 20 clientes VM Maestro, implicaría la necesidad de más recursos de hardware para el software, puesto que se debe mantener más nodos simulados de manera simultánea, debido al aumento de proyectos y usuarios.

Bibliografía

[1] S. K. E. K. Q. S. W. T. B. D. B. Joel Obstfeld, "VIRL: Virtual Internet Routing Lab.," in *SIGCOMM*, Chicago, Illinois, USA, 2014.

[2] «wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Cisco_Systems.

[3] «Virtual Internet Routing Lab (VIRL),» [En línea]. Available: https://learningnetwork.cisco.com/groups/virl.

[4] «Cisco VIRL,» [En línea]. Available: http://virl.cisco.com/.

[5] Z. M. a. J. V. d. M. X. Chen, «ShadowNet: a platform for rapid and safe network evolution,» de *USENIX Annual technical conference*, 2009.

[6] J. C. Contreras, «Evaluación de plataformas de virtualización para simular o emular redes convergentes,» Valparaíso, 2015.

[7] A. W. Matias Comba, *Diseño e implementación de laboratorios virtuales multimarca,* 2014.

[8] «wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/Dynamips.

[9] «Qemu,» [En línea]. Available: http://www.qemu-project.org/.

[10] «virtualbox,» [En línea]. Available: https://www.virtualbox.org/.

[11] «wireshark,» [En línea]. Available: https://www.wireshark.org/.

[12] «Wiki freecode,» [En línea]. Available: https://wiki.freecode.com.cn/doku.php?id=wiki:vpcs.

[13] «nongnu,» [En línea]. Available: http://www.nongnu.org/quagga/.

[14] «cisco packet tracer,» [En línea]. Available: https://www.netacad.com/es/web/about-us/cisco-packet-tracer.

[15] «Mininet,» [En línea]. Available: http://mininet.org/.

[16] «Netkit,» [En línea]. Available: http://wiki.netkit.org/.

[17] M. M. R. C. Q. J. C. J. D. T. V. Manuel A. Calle Pérez, «Simulando con OMNET selección de la herramienta y su utilización,» Universidad ICESI, 2013.

[18] «Administración de sistemas operativos,» [En línea]. Available: http://www.adminso.es/.

[19] «VMware,» [En línea]. Available: https://www.vmware.com/.

[20] «GNS3,» [En línea]. Available: https://www.gns3.com/software.

[21] «Propiedad Intelectual,» [En línea]. Available: https://es.wikipedia.org/wiki/Propiedad_intelectual.

[22] E. V. y. J. Gómez, «Intoroducción a la Virtualización,» de *VIRTUALIZACION DE SERVIDORES DE TELEFONIA IP EN GNU/LINUX*.

[23] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/VMware.

[24] «VMware Workstation 12.5 Pro,» [En línea]. Available: http://store.vmware.com/store/vmware/en_US/DisplayProductDetailsPage/ThemeID.2485600/productID.323700100?src=WWW_eBIZ_productpage_Workstation_Buy_US.

[25] a.-V. Support, «The Cisco Learning Network,» Abril 2016. [En línea]. Available: https://learningnetwork.cisco.com/docs/DOC-30230.

[26] «Cisco VIRL,» [En línea]. Available: http://virl-dev-innovate.cisco.com/.

[27] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Open_Shortest_Path_First.

[28] «REDESCISCO.NET,» [En línea]. Available: http://www.redescisco.net/sitio/2013/08/11/cisco-asa-configurando-interfaces/.

[29] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Transferencia_de_Estado_Representacional.

[30] J. Wang, The VIRL Book, 2016.

[31] «Certificaciones,» [En línea]. Available: http://www.cisco.com/c/es_mx/training-events/certifications.html.

[32] «packettracernetwork,» [En línea]. Available: http://www.packettracernetwork.com/.

[33] Valerie, «The Cisco Learning Network,» 2016. [En línea]. Available: https://learningnetwork.cisco.com/docs/DOC-30476.

A Apéndice

A.1 Costo solución Cisco VIRL

Se estudia el costo para la implementación de un laboratorio de redes con Cisco VIRL, con capacidad para 20 estaciones de trabajo.

Consideraciones:

- ✓ Las cotizaciones fueron realizadas en las empresas PC Factory, Sodimac, IPEX (https://www.ipexpress.cl/) y NETCO (http://www.netco.cl/).
- ✓ El costo de la licencia corresponde a un desembolso que debe realizarse una vez por año (cada 12 meses).
- ✓ El costo de adquisición de VIRL Academic license fue de $79.99 UDS. Este valor ya no se encuentra disponible, sino solo la edición personal. Por tanto la renovación de la licencia corresponde al valor mostrado en el punto 6 de la Tabla A-1.
- ✓ Costo servicio de instalación corresponde al 65% del costo bruto en equipamiento para la instalación y puesta en marcha de Cisco VIRL, y un 30% para la instalación de equipamiento real.

Tabla A-1. Detalle costo para la implementación de laboratorio con software Cisco VIRL.

Producto	Precio unitario efectivo (IVA incluido)	Cantidad	Sub Total
1. PC tipo desktop			
S. Técnico Servicio Armado 2 - Test y configuración	$ 12.890	1	$ 12.890
Intel CPU Core i5-4460 3.2 GHz (1150)	$ 169.990	1	$ 169.990
MSI M/B Intel® Z97 PC Mate (1150)	$ 99.990	1	$ 99.990
Corsair DDR3 8GB 1333MHZ PC3-10666 XMS3	$ 49.990	2	$ 99.980
Crucial Unidad SSD 480GB BX200 Sata3 2.5"	$ 97.890	1	$ 97.890

Spektra Fuente Poder 500W 12cm	$ 18.990	1	$ 18.990
AOC Monitor LED 20" E2070SWN	$ 64.990	1	$ 64.990
Genius Mouse DX-110 Negro	$ 4.690	1	$ 4.690
Genius Teclado USB KB-M200 Negro	$ 6.590	1	$ 6.590
Cougar Gabinete ATX MX200 5VS9 s/F	$ 28.990	1	$ 28.990
Spektra Cable SATA (Serial-Ata) Datos (7P) 90º	$ 1.990	1	$ 1.990
2. Switch			
TP-Link Switch 24b TL-SF1024 10/100	$ 39.990	1	$ 39.990
3. PC tipo desktop básico			
AIO S200Z Celeron N3050 4GB 500GB 19,5" HD+ Free DOS	$ 252.190	20	$ 5.043.800
4. Cableado estructurado			
Rollo Cable UTP Cat6 305m	$ 72.990	1	$ 72.990
Conector RJ45 CAT6 10 unidades	$ 1.990	5	$ 9.950
Canaleta 100X50mm 2mts blanca	$ 10.850	10	$ 108.500
5. Estación de trabajo			
Escritorio	$ 24.990	20	$ 499.800
Silla	$ 21.990	20	$ 439.800
Mouse pad	$ 4.690	20	$ 93.800
6. Softwares.			
Microsoft Windows 7 Professional OEM 64 bit	$ 124.990	1	$ 124.990
VMware Workstation Pro 12	$ -	1	$ -
VIRL Personal Edition 20-Node (cada 12 meses)	$ 134.245	1	$ 134.245
7. Servicios.			
Servicios de instalación y puesta en marcha	$ 4.663.649	1	$ 4.663.649
		TOTAL NETO	$ 9.948.314
		IVA	$ 1.890.180
		TOTAL	$ 11.838.494

El costo total bruto para la implementación de un laboratorio con Cisco VIRL asciende a la suma de $ 11.838.494 CLP.

A.2 Costo solución equipamiento real

Se realiza un análisis de costo para la implementación del laboratorio con equipamiento real. Con un total de 8 routers, 3 switchs, un ASA y 5 estaciones de trabajo.

Tabla A-2. Detalle costo para la implementación de laboratorio con equipamiento real.

Producto	Precio unitario efectivo (IVA incluido)	Cantidad	Sub Total
1. Router			
Router Cisco CISCO1921-SEC/K9	$ 1.040.631	6	$ 6.243.787
Router Cisco CISCO2901-SEC/K9	$ 1.995.487	3	$ 5.986.462
2. Switch			
Switch Cisco WS-C2960S-24TS-S	$ 1.270.444	1	$ 1.270.444
Switch Cisco WS-C2960-24TC-S	$ 506.559	2	$ 1.013.118
3. ASA			
UTM Cisco ASA5508-K9	$ 1.823.937	1	$ 1.823.937
4. PC básico + Estación de trabajo			
AIO S200Z Celeron N3050 4GB 500GB 19,5" HD+ Free DOS	$ 252.190	5	$ 1.260.950
Escritorio	$ 24.990	5	$ 124.950
Silla	$ 24.990	5	$ 124.950
Mouse pad	$ 4.690	5	$ 23.450
5. Cableado estructurado			
Rollo Cable UTP Cat6 305m	$ 72.990	1	$ 72.990
Conector RJ45 CAT6 10 unidades	$ 1.990	6	$ 11.940
Cisco CAB-CONSOLE-RJ45= (spare)	$ 28.322	5	$ 141.610
Cisco CAB-CONSOLE-USB= (spare)	$ 28.322	5	$ 141.610
Cisco CAB-SS-232FC= (spare)	$ 76.065	5	$ 380.324
PDU REGLETA 9 ENCHUFES CH 10A RACK	$ 40.840	2	$ 81.680
6. Rack			
Rack 60" 32U 600x800	$ 289.990	1	$ 289.990
7. Servicios			
Servicio de instalación	$ 5.697.658	1	$ 5.697.658
		TOTAL NETO	$ 20.747.773
		IVA	$ 3.942.077
		TOTAL	$ 24.689.850

El costo total bruto para la implementación de un laboratorio con equipamiento real, asciende a la suma de $24.689.850.

A.3 Glosario de términos.

- **CDP** (Cisco Discovery Protocol, 'protocolo de descubrimiento de Cisco'), es un protocolo de red propietario de nivel 2, desarrollado por Cisco Systems y usado en la mayoría de sus equipos.
- **LXC** (Linux Containers) es una tecnología de virtualización en el nivel de sistema operativo para Linux. LXC permite que un servidor físico ejecute múltiples instancias de sistemas operativos aislados, conocidos como Servidores Privados Virtuales (SPV o VPS en inglés) o Entornos Virtuales. LXC no provee de una máquina virtual, más bien provee un entorno virtual que tiene su propio espacio de procesos y redes.
- **SSH** (Secure SHell, en español: intérprete de órdenes seguro) protocolo que sirve para acceder a máquinas remotas a través de una red.
- **Telnet** (Telecommunication Network1) protocolo de red que permite viajar a otra máquina para manejarla remotamente como si se estuviera sentado delante de ella.
- **RSA** (Rivest, Shamir y Adleman) es un sistema criptográfico de clave pública. Es el primer y más utilizado algoritmo de este tipo y es válido tanto para cifrar como para firmar digitalmente.
- **OSPF** (Open Shortest Path First), camino más corto primero, es un protocolo de red para encaminamiento jerárquico de pasarela interior o Interior Gateway Protocol (IGP), que usa el algoritmo SmoothWall Dijkstra enlace-estado (Link State Advertisement, LSA) para calcular la ruta idónea entre dos nodos cualesquiera de un sistema autónomo.
- **IGP** (Interior Gateway Protocol), Protocolo de Pasarela Interna o Protocolo de Pasarela Interior, hace referencia a los protocolos usados dentro de un sistema autónomo.
- **iBGP eBGP**: La forma de configurar y delimitar la información que contiene e intercambia el protocolo BGP, es creando lo que se conoce como sistema autónomo. Cada sistema autónomo tendrá, sesiones internas (iBGP) y además sesiones externas (eBGP).
- **Route-reflector** es un Router que envía actualizaciones a routers clientes. Cuando un cliente envía una actualización al route-reflector, es enviado a su vez o reflectado a otros clientes.
- **Peering** (intercambio de tráfico o emparejamiento) es la interconexión voluntaria de redes de Internet administrativamente independientes con el fin de intercambiar tráfico de los clientes de cada red.
- **IS-IS** (del inglés Intermediate system to intermediate system) es un protocolo de estado de enlace, o SPF (shortest path first), por lo cual, básicamente maneja una especie de mapa con el que se fabrica a medida que converge la red. Es también un protocolo de Gateway interior (IGP).
- **EIGRP** (en español: Protocolo de Enrutamiento de Puerta de enlace Interior Mejorado) es un protocolo de encaminamiento vector distancia avanzado, propiedad de Cisco Systems, que ofrece lo mejor de los algoritmos de vector de distancias y del estado de enlace.
- **NAT** (del inglés Network Address Translation) es un mecanismo utilizado por routers IP para intercambiar paquetes entre dos redes que asignan mutuamente direcciones incompatibles. Consiste en convertir, en tiempo real, las direcciones utilizadas en los

paquetes transportados. También es necesario editar los paquetes para permitir la operación de protocolos que incluyen información de direcciones dentro de la conversación del protocolo.

- **OpenStack** es un proyecto de computación en la nube para proporcionar una infraestructura como servicio (IaaS).
  Es un software libre y de código abierto distribuido bajo los términos de la licencia Apache. El proyecto está gestionado por la Fundación OpenStack, una persona jurídica sin fines de lucro creada en septiembre de 2012 para promover el software OpenStack y su comunidad.
- **IaaS** (*infrastructure as a service*, infraestructura como un servicio) se encuentra en la capa inferior y es un medio de entregar almacenamiento básico y capacidades de cómputo como servicios estandarizados en la red. Servidores, sistemas de almacenamiento, conexiones, enrutadores, y otros sistemas se concentran (por ejemplo a través de la tecnología de virtualización) para manejar tipos específicos de cargas de trabajo, desde procesamiento en lotes ("batch") hasta aumento de servidor/almacenamiento durante las cargas pico.
- **Nova** es un controlador de estructura cloud computing, que es la parte principal de un sistema de IaaS. Está diseñado para gestionar y automatizar los pools de los recursos del equipo y puede trabajar con tecnologías ampliamente disponibles de virtualización. KVM y Xen son las opciones disponibles para la tecnología de hipervisor, junto con la tecnología Hyper-V, la tecnología vSphere de VMware y la tecnología de contenedores Linux como LXC.
- **Neutron** es un sistema para la gestión de redes y direcciones IP. Asegura que la red no presente el problema del cuello de botella o el factor limitante en un despliegue en la nube y ofrece a los usuarios un autoservicio real, incluso a través de sus configuraciones de red.
- **Glance** proporciona servicios de descubrimiento, de inscripción y de entrega de los discos y del servidor de imágenes. Las imágenes almacenadas se pueden utilizar como una plantilla. También se puede utilizar para almacenar y catalogar un número ilimitado de copias de seguridad.
- **Keystone** es el servicio de Identidad de OpenStack y ofrece un directorio central de usuarios asignados a los servicios de OpenStack que pueden acceder. Actúa como un sistema de autenticación común en todo el sistema operativo para la nube.
- **URI** (Uniform Resource Identifier o en español identificador de recursos uniforme) es una cadena de caracteres que identifica los recursos de una red de forma unívoca.
- **URL** (Uniform Resource Locator o en español localizador de recursos uniforme) es un identificador de recursos uniforme (URI) cuyos recursos referidos pueden cambiar, esto es, la dirección puede apuntar a recursos variables en el tiempo. Están formados por una secuencia de caracteres, de acuerdo a un formato modélico y estándar, que designa recursos en una red, como Internet.
- **HTTP** (Hypertext Transfer Protocol o en español protocolo de transferencia de hipertexto) es el protocolo de comunicación que permite las transferencias de información en la World Wide Web. HTTP define la sintaxis y la semántica que utilizan

los elementos de software de la arquitectura web (clientes, servidores, proxies) para comunicarse.

- **Jitter** (término inglés para fluctuación). Es la variabilidad temporal durante el envío de señales digitales como una ligera desviación de la exactitud de la señal de reloj. El jitter suele considerarse como una señal de ruido no deseada. En general se denomina jitter a un cambio indeseado y abrupto de la propiedad de una señal. Esto puede afectar tanto a la amplitud, la frecuencia y la fase de la señal.
- **QoS** (*Quality of Service*, Calidad de Servici), es el rendimiento promedio de una red de telefonía o de computadoras, particularmente el rendimiento visto por los usuarios de la red.
- **FPGA** (*Field Programmable Gate Array*) es un dispositivo programable que contiene bloques de lógica cuya interconexión y funcionalidad puede ser configurada "in situ" mediante un lenguaje de descripción especializado.
- **Telnet** (Telecommunication Network1) protocolo de red que permite viajar a otra máquina para manejarla remotamente como si se estuviera sentado delante de ella.
- **RSA** (Rivest, Shamir y Adleman) es un sistema criptográfico de clave pública. Es el primer y más utilizado algoritmo de este tipo y es válido tanto para cifrar como para firmar digitalmente.