ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

ANÁLISIS DE PROTOCOLOS PARA LA GESTIÓN DE REDES

ANÁLISIS DEL PROTOCOLO DE GESTIÓN DE RED NETCONF

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TELECOMUNICACIONES

ALEX DANIEL GUAMÁN PACHACAMA

alex.guaman01@epn.edu.ec

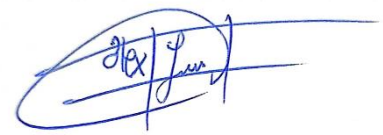
DIRECTOR: M.Sc. XAVIER ALEXANDER CALDERÓN HINOJOSA

xavier.calderon@epn.edu.ec

DMQ, octubre 2022

CERTIFICACIONES

Yo, Alex Daniel Guamán Pachacama declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.



Alex Daniel Guamán Pachacama

Certifico que el presente trabajo de integración curricular fue desarrollado por Alex Daniel Guamán Pachacama, bajo mi supervisión.

M.Sc. Xavier Alexander Calderón Hinojosa

DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

ALEX DANIEL GUAMÁN PACHACAMA

M.SC. XAVIER ALEXANDER CALDERÓN HINOJOSA

DEDICATORIA

El presente trabajo es el fruto del esfuerzo constante y la superación de retos que he afrontado a lo largo de mi formación académica en la prestigiosa Escuela Politécnica Nacional, es por esto que lo dedico a mis queridos padres Luis y Verónica; y a mi hermano Adrián, quienes con su cariño, consejos, apoyo constante y confianza me han impulsado a cumplir uno más de mis objetivos personales.

También va dedicado a mis buenos amigos Daniel, Francisco, André, Erick, Kathy, Carol y Alejandra quienes con su aliento constante, apoyo desinteresado y buenos consejos me han motivado a alcanzar una meta más en mi vida. Es por ellos y sobre todo por los momentos de alegrías y tristezas que hemos vivido juntos que el camino que he recorrido hasta ahora ha sido una aventura inolvidable.

Alex Guamán

AGRADECIMIENTO

En primer lugar, quiero dar gracias a Dios y a mi madre Dolorosa por permitirme estudiar en una de las mejores universidades del país, gracias a mi hermosa familia por siempre apoyarme y darme los ánimos que necesite en este arduo camino. Especialmente quiero agradecer a mis padres por todas sus enseñanzas y valores que me han sabido inculcar, por ellos he llegado a ser el hombre que soy ahora.

También quiero agradecer a mis amigos más cercanos, especialmente a mis queridos amigos del Colegio San Gabriel que, a pesar de haber tomado caminos diferentes, nunca han dejado de apoyarme y ofrecerme su ayuda.

Finalmente quiero dar gracias a todo el personal docente y compañeros que he tenido el gusto de conocer en mi querida Poli, los cuales han aportado tanto en mi crecimiento personal como académico.

ÍNDICE DE CONTENIDO

RESUMEN

Hoy en día, todo tipo de organización dispone de alguna red de telecomunicaciones ya que facilita la comunicación de sus miembros y colaboradores. Debido a la creciente demanda de los usuarios dichas redes se han visto obligadas a mejorar constantemente, ocasionando una mayor dificultad en la administración de las mismas. El protocolo SNMP y la CLI son las tecnologías que actualmente son usadas para la gestión de redes, sin embargo, estas no representan una forma eficaz y escalable de configurar y administrar los dispositivos de red de las nuevas redes emergentes. El protocolo NETCONF surge como una solución a la problemática anteriormente mencionada, razón por la cual administradores de red e incluso proveedores de software y equipos de red, han adoptado rápidamente dicho protocolo para la administración de redes. NETCONF apunta a ser uno de los protocolos de gestión de red más utilizados en un futuro próximo debido a sus ventajas, es por esta razón que aprender sobre NETCONF resulta ser una idea inteligente. Este trabajo propone y contiene un total de 5 prácticas de laboratorio elaboradas de acuerdo al formato establecido por el Departamento de Electrónica, Telecomunicaciones y Redes de Información de la Escuela Politécnica Nacional, las cuales permiten demostrar las características del protocolo NETCONF en un entorno emulado de GNS3 entre cliente y servidor. Cada una de las prácticas de laboratorio consta de actividades que permitirán a los estudiantes de la carrera de Ingeniería en Telecomunicaciones y afines comprender, aprender y aplicar las funcionalidades que ofrece NETCONF.

PALABRAS CLAVE: Prácticas de laboratorio, NETCONF, YANG, servidor, cliente, GNS3.

ABSTRACT

Nowadays, every type of organization has some kind of telecommunications network, since it facilitates communication among its members and collaborators. Due to the growing demand of users, these networks have been forced to improve constantly, causing a greater difficulty in their administration. The SNMP protocol and CLI are the technologies that are currently used for network management; however, they do not represent an efficient and scalable way to configure and manage network devices of the new emerging networks. The NETCONF protocol emerges as a solution to the aforementioned problems, which is why network administrators and even software and network equipment vendors have been quick to adopt this protocol for network management. NETCONF aims to be one of the most widely used network management protocols in the near future due to its benefits, which is why learning about NETCONF is a smart idea. This work proposes and contains a total of 5 laboratory practices elaborated according to the format established by the Department of Electronics, Telecommunications and Information Networks of Escuela Politécnica Nacional, which allow to demonstrate the characteristics of the NETCONF protocol in an emulated environment of GNS3 between client and server. Each of the laboratory practices consists of activities that will allow students of Telecommunications Engineering and related fields to understand, learn and apply the functionalities offered by NETCONF.

KEYWORDS: Lab Practices, NETCONF, YANG, server, client, GNS3.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

Hoy en día las redes de Telecomunicaciones desempeñan un papel primordial en la sociedad, esto debido a que facilitan la comunicación a las personas y a las organizaciones de las que son parte. Ahora es muy común encontrar empresas que dispongan de algún tipo de red, ya que son esenciales para mejorar la comunicación con colaboradores estratégicos, y por tanto el incremento de la productividad y de los ingresos.

Actualmente las redes están en constante cambio y crecimiento, esto debido a la gran demanda por parte de los usuarios. Esto quiere decir que se necesitan redes más potentes y capaces de satisfacer todas las exigencias de los mismos. Sin embargo, una mejora en la red también involucra una mayor complejidad de administración. Hoy por hoy, se emplea el protocolo SNMP (Simple Network Management Protocol) y la CLI (Command Line Interface) para el monitoreo y la configuración de los dispositivos de red. No obstante, ambas están quedando obsoletas debido a que no se ajustan a la necesidad de escalabilidad para la gestión de red [1]. Por un lado, usar el protocolo SNMP limita la administración ya que no permite realizar configuraciones de red complejas, razón por la cual está enfocada solo en el monitoreo de red [1]. Por el otro, la interfaz CLI a pesar de que permite realizar configuraciones de red no cuenta con respuestas estructuradas para errores, además no posee un estándar [2], lo que implica que cada proveedor maneja sus propios conjuntos de comandos y procedimientos patentados para la configuración de dispositivos [3], dificultando en gran medida la administración de la red.

Debido a la problemática anteriormente mencionada nace el protocolo NETCONF (Network Configuration Protocol), mostrándose como una solución óptima para los clientes e incluso proveedores. Esto debido a que ofrece una interfaz de programación estándar para la configuración y gestión de la red, que aborda muchos de los desafíos del protocolo SNMP y la interfaz CLI [4]. NETCONF ha sido adoptada rápidamente por muchos proveedores y clientes. Actualmente este protocolo continúa evolucionando y ganando mayor aceptación [4], razón por lo cual ya se encuentra presente en varios dispositivos de red de distintos proveedores de software y equipos de red.

Lo que se quiere lograr con este trabajo de titulación es estudiar el protocolo de gestión de red NETCONF, así también se quiere demostrar su utilidad, por lo cual se plantea realizar una serie de prácticas de laboratorio, las cuales estarán dirigidas a los estudiantes de la carrera de Telecomunicaciones y afines. Gracias a estas prácticas los estudiantes serán capaces de utilizar dichos conocimientos en un futuro cercano, logrando así destacar en el campo laboral.

1.1 Objetivo general

Analizar el protocolo de gestión de red NETCONF utilizando simuladores/emuladores de equipos de red.

1.2 Objetivos específicos

1. Analizar los fundamentos teóricos del protocolo de gestión de red NETCONF.
2. Implementar prácticas de laboratorio empleando una topología de red en GNS3 que sea idónea para explicar los fundamentos teóricos referentes al protocolo NETCONF.
3. Analizar los resultados obtenidos al cumplir el procedimiento de cada una de las prácticas de laboratorio.

1.3 Alcance

El trabajo de Titulación estará enfocado en primera instancia en el análisis del protocolo NETCONF, donde se detallarán las diferencias frente a SNMP, su funcionamiento, se explicarán sus cuatro capas, sus operaciones base y sus capacidades. Así también se detallará el modelo de datos YANG y su relación con el protocolo NETCONF. Además, se realizará una lista de los equipos que tienen la capacidad de soportar la configuración de este protocolo; de tres proveedores del área de Redes como lo son Cisco, Juniper y Huawei.

Luego el trabajo se enfocará en la elaboración de 5 prácticas de laboratorio que incluirán una serie de elementos a elaborar como son: preparatorios, informes y guías de configuración. Estas prácticas estarán dirigidas para los estudiantes de la carrera de Telecomunicaciones y afines, esto con el propósito de que los mismos sean capaces de aplicar y emplear las características que ofrece el protocolo NETCONF en un futuro cercano.

Este trabajo de Titulación, constará de 3 etapas, las cuales se detallan a continuación:

A. Fase de planteamiento

En esta fase se estudiará acerca de los fundamentos teóricos del protocolo de gestión de red NETCONF. Se estudiará específicamente sus cuatro capas, el funcionamiento, las diferencias frente a SNMP, las operaciones base, las capacidades y la relación de NETCONF con XML y el modelo de datos YANG. Además, se realizará una lista de los equipos que soportan dicho protocolo de tres proveedores del área de Redes como los son Cisco, Juniper y Huawei.

También en esta fase, se empezará a realizar pequeñas pruebas en el emulador GNS3, donde se instalarán distintas imágenes de los dispositivos (routers y switches) capaces de soportar el protocolo NETCONF, así también se probarán distintas máquinas virtuales, las cuales permitan simular un cliente NETCONF. Al final de esta fase se escogerá un router o switch y una máquina virtual con las cuales se trabajará para realizar las 5 prácticas de laboratorio. Los temas previstos de estas prácticas, tendrán las siguientes temáticas.

- Práctica 1: Introducción a NETCONF.
- Práctica 2: Capa de transporte seguro de NETCONF.
- Práctica 3: Lenguaje de modelado de datos YANG.
- Práctica 4: Operaciones y capacidades NETCONF
- Práctica 5: Cliente NETCONF con Python.

B. Fase de implementación

Esta fase tendrá como propósito el realizar las 5 prácticas de laboratorio, para ello se realizará las correspondientes emulaciones en GNS3. Se probará el funcionamiento de cada una las prácticas, se documentará lo realizado, se realizará un video explicativo y se propondrán los enunciados para la parte del preparatorio, informe y coloquio.

C. Fase de análisis de resultados

Se redactará el documento final basado en el formato IEEE con la información utilizada. Además, se escribirán las 5 prácticas con el formato establecido por el Departamento de Electrónica, Telecomunicaciones y Redes de Información. Por último, se detallarán las conclusiones y recomendaciones obtenidas en el trabajo de Titulación.

1.4 Marco teórico

1.4.1 Descripción general del protocolo NETCONF

1.4.1.1 Pequeña reseña histórica

El requisito de tener una manera eficaz y escalable de configurar y administrar dispositivos de red existe ya desde hace mucho tiempo atrás. SNMP apareció por primera vez a finales de los años 80 y fue diseñado por la IETF (Internet Engineering Task Force) para brindar una interfaz estándar que permita la configuración y monitoreo de la información de administración de los elementos de red [5]. Aunque dicho protocolo fue evolucionando a lo largo de los años, no logró adaptarse a las necesidades de las nuevas redes emergentes, debido a que presentaba algunos inconvenientes. SNMP al hacer uso del protocolo UDP

(User Datagram Protocol) para transferir mensajes, proporcionaba una falta de seguridad en la comunicación de los dispositivos de red, ya que al no establecer una conexión segura lo hacía vulnerable a ataques informáticos externos [1]. Así también, SNMP no realiza ninguna distinción entre información de estado y de configuración [6] lo que complicaba en gran medida la gestión de red. Por último, SNMP no resultó ser útil para realizar configuraciones complejas y especializada en los dispositivos de red, razón por la cual su uso fue limitado solo para el monitoreo de red [1].

Debido a que SNMP no cumplió las expectativas para la configuración de dispositivos de red, se hizo necesario emplear la CLI para satisfacer este requerimiento. Sin embargo, esto tampoco resultó ser lo más viable. Emplear la CLI requiere que los administradores de red aprendan varios comandos de configuración para dispositivos de diferentes proveedores [3], lo que implica altos costos de operación y mantenimiento en las redes. Debido a que la CLI no posee un estándar, no se ha logrado establecer un lenguaje de descripción formal para definir todas las propiedades de una interfaz de programación. Esto implica que la automatización de la configuración de los dispositivos de red por medio de programas y scripts resulte ser problemática, especialmente cuando se presentan problemas de control de versiones en la CLI [6]. Por último, la CLI no posee ninguna estructura de respuestas de errores [3] lo que limita en gran medida la administración de la red.

Para el año 2002, la IETF organizó un taller para considerar el estado actual de la gestión de redes, al cual acudieron proveedores de redes, operadores, proveedores de servicios y más. El resultado de dicho taller fue el RFC 3535, en donde se establecieron recomendaciones para la próxima evolución de los protocolos de gestión de redes [4]. Entre ellas se destacan la creación de una interfaz de programación para la configuración de dispositivos, la separación de datos de estado y de configuración, la capacidad de configurar servicios y no dispositivos, la comprobación y recuperación de errores, entre otras. En base a los requerimientos establecidos en el RFC 3535, la IETF desarrolló NETCONF propuesto en el RFC 4741 para el año 2006 y YANG en el RFC 6020 para el 2010. Logrando así ofrecer a proveedores y administradores de red un protocolo estándar y un lenguaje de modelado de datos para la gestión de redes programáticas, respectivamente.

1.4.1.2 Definición del protocolo

NETCONF es un protocolo de gestión de red que define un mecanismo para la administración de dispositivos de red, con el que se puede recuperar tanto información de configuración como de estado, así como cargar y manipular datos de configuración [7]. Uno de los mayores beneficios al usar el protocolo NETCONF es que ofrece a los dispositivos

una completa y formal API (Application Programming Interface), la cual puede ser usada por aplicaciones para enviar y recibir conjuntos de datos de configuración completos y parciales [7]. Permitiendo de esta forma el uso de la programación como por ejemplo Python, para gestionar las redes.

El protocolo NETCONF obliga el uso de protocolos de transporte seguros como lo es SSH (Secure Shell) para comunicación entre cliente y servidor. Brindando de esta manera una conexión segura para el intercambio de información.

NETCONF hace uso de RPC (Remote Procedure Call) y XML (Extensible Markup Language) para el intercambio de información entre cliente y servidor. Por un lado, RPC establece el modelo de comunicación entre cliente y servidor, mientras que XML ofrece el formato y la codificación de los datos intercambiados [7].

Un aspecto importante de NETCONF es que permite que la funcionalidad del protocolo siga de cerca la funcionalidad nativa del dispositivo. Esto implica que usuarios y/o aplicaciones serán capaces de acceder tanto al contenido sintáctico y sistemático de las interfaces de usuarios de los dispositivos. Permitiendo de esta forma el surgimiento de nuevas funcionalidades, así como la reducción de costos de implementación [7].

Por último, NETCONF permite que los clientes descubran las funcionalidades que son soportadas por el servidor. A estas se las denomina “Capacidades” y son aquellas que permiten al cliente tomar ventaja de las mismas para mejorar el rendimiento de la administración del dispositivo [7].

1.4.1.3 Funcionamiento

NETCONF emplea un mecanismo basado en RPC para la comunicación entre un cliente y un servidor. El cliente puede ser un script o aplicación ejecutada comúnmente por un administrador de red, mientras que el servidor suele ser un dispositivo de red. El proceso de comunicación inicia al establecer una conexión lógica entre cliente y servidor que debe ser segura y orientada a la conexión. A esto se lo denomina sesión NETCONF y se consigue después del intercambio de capacidades entre servidor y cliente mediante mensajes <hello>. Continuando con el proceso, el cliente enviará un mensaje RPC codificado y en formato XML al servidor. Luego el servidor responderá con una réplica codificada de igual manera en XML. Por último, se tendrá que cerrar la sesión NETCONF para así finalizar la comunicación [7].

NETCONF está dividido conceptualmente en cuatro capas, como se muestra en la Figura 1.1

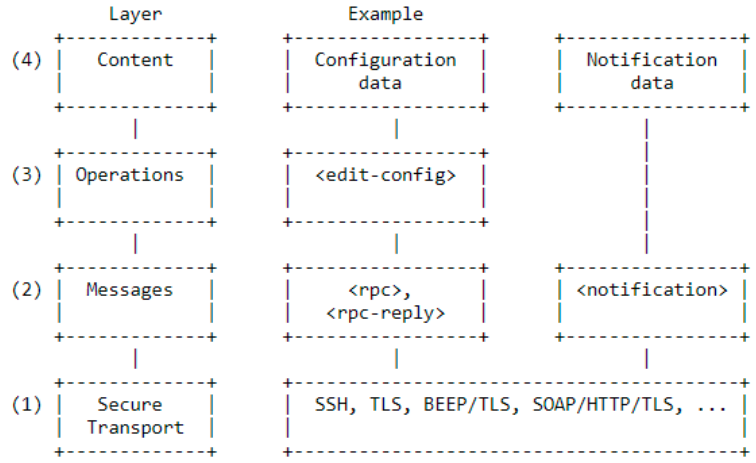


Figura 1.1 Arquitectura de capas del protocolo NETCONF [7].

1. Capa de Transporte Seguro: Proporciona una ruta de comunicación segura entre cliente y servidor, la cual debe cumplir con un conjunto de requisitos.

2. Capa de Mensajes: Provee un mecanismo/estándar de estructuración de tramas simples para codificar RPC y notificaciones. Los mensajes RPC se documentan en RFC 6241 y las notificaciones en RFC 5717.

3. Capa de Operaciones: Establece un conjunto de operaciones base del protocolo NETCONF, las mismas que son invocadas como métodos RPC codificados en XML.

4. Capa de Contenido: El RFC 6241 no define la capa de contenido. Sin embargo, el lenguaje de modelado de datos YANG definido en el RFC 6020 fue desarrollado para especificar los modelos de datos NETCONF y las operaciones del protocolo, cubriendo de esta forma la capa de contenido y la capa de operaciones.

1.4.1.4 Comparación con SNMP

A razón de entender de mejor manera las diferencias entre el protocolo SNMP y NETCONF, se realizó la Tabla 1.1 tomando en cuenta [8] y [9]

Tabla 1.1 Comparación entre NETCONF y SNMP

Característica	NETCONF	SNMP
Legible por humanos	Si	No
Estándar	Si, por parte de IETF.	Si, por parte de IETF.
Protocolo de Transporte seguro	Si, por medio de SSH, TLS, etc.	No, en las versiones 1 y 2c. En versión 3 ofrece TLS y DTLS
Distinción entre datos de configuración y estado.	Si	No
Acceso a recursos	Por medio de paths	Por medio de OIDs

Modelo de datos	Modelos YANG	Definidos en MIBs
Lenguaje de modelado de datos	YANG	SMI
Codificación	XML	BER
Envió de notificaciones de eventos	Si, por medio de suscripción.	Si
Configuración de múltiples elementos de la red basada en operaciones.	Si	No
Configuración de copia de seguridad y restauración.	Si	No
Probar configuración antes de confirmación final.	Si	No
Usos	Para configuración (automatizada) y monitoreo de red.	Para monitoreo de red.

1.4.2 Capa de transporte seguro

NETCONF se puede superponer sobre cualquier protocolo de transporte que proporcione un mecanismo para indicar el tipo de sesión (cliente o servidor) a la capa de transporte seguro. En sí, NETCONF no está vinculado a un protocolo de transporte en particular, pero define ciertos requerimientos [7], los cuales se detallan a continuación.

1. Orientado a la conexión

El protocolo de transporte escogido debe ofrecer una operación orientada a la conexión. En primer lugar, porque NETCONF necesita una comunicación permanente en pares (cliente y servidor). Segundo, la conexión debe ser capaz de entregar los datos de forma confiable y secuencial. Por último, los recursos solicitados al servidor en la conexión deben ser liberados una vez finalizada la conexión, permitiendo que la recuperación de fallos sea simple y robusta [7].

2. Autenticación, integridad y confidencialidad

El protocolo de transporte escogido debe permitir que las conexiones NETCONF ofrezcan autenticación, integridad de datos, confidencialidad y protección de repetición (Replay Protection). Esto quiere decir que cada par NETCONF se rige a los mecanismos de seguridad, confidencialidad y autenticación establecidos por el protocolo de transporte [7].

Debido a que el protocolo de transporte es el encargado de realizar la autenticación del servidor al cliente y del cliente al servidor. Lo único que realiza cada par NETCONF, es asumir que dicho proceso de autenticación es lo suficientemente confiable ya que fue

validado por el protocolo de transporte. El resultado del proceso de autenticación es una identidad del cliente autenticado que se denomina usuario NETCONF. Esto quiere decir que el protocolo de transporte es el responsable de proporcionar un usuario NETCONF para que pueda ser usado por las otras capas de NETCONF [7].

Gracias a que NETCONF sigue de cerca la funcionalidad nativa de los dispositivos, el protocolo de transporte es capaz de emplear los mecanismos de autenticación disponibles en el dispositivo sin ningún problema. Por ejemplo, se puede usar un servidor TACACS+ para autenticar sesiones NETCONF [7].

1.4.2.1 Protocolo de transporte obligatorio

A pesar de que NETCONF no define un protocolo de transporte específico a utilizar, el protocolo obligatorio de transporte para una implementación de NETCONF es SSH [7]. Una implementación NETCONF con SSH se puede apreciar en el RFC 6242.

1.4.3 XML

XML sirve como formato de codificación para el protocolo NETCONF. Su función es establecer una jerarquización compleja de datos, que sea capaz de expresarse en un formato de texto que pueda ser leído, guardado y manipulado ya sea con herramientas de texto tradicionales o con herramientas específicas de XML [7].

XML se encuentra presente en tres de las cuatro capas del protocolo NETCONF. Su sintaxis, así como su codificación es empleada en las RPC de la capa de mensajes, en las operaciones del protocolo y en los módulos YANG de la capa de contenido [7] [10].

Es necesario mencionar que todos los mensajes de NETCONF deben estar correctamente estructurados en XML y codificados en UTF-8. Si se envía un mensaje RPC que no se encuentra bien estructurado en XML o no codificado en UTF-8, se deberá responder con un error de “mensaje mal formado” [7].

1.4.3.1 Espacio de nombres

En XML los nombres de los elementos y atributos pertenecen a un espacio de nombres, el cual es identificado mediante una referencia URI (Uniform Resource Identifiers) [11]. En NETCONF todos los elementos del protocolo se definen en el siguiente espacio de nombres [7] :

urn:ietf:params:xml:ns:netconf:base:

Los modelos de datos YANG también poseen un espacio de nombres, lo que indica que poseen sus propios elementos y atributos.

1.4.4 Capa de mensajes: Modelo RPC

NETCONF emplea un modelo de comunicación basado en RPC, en donde los pares (cliente y servidor) usan elementos <rpc> y <rpc-reply> para el intercambio de mensajes de solicitud y respuesta respectivamente. Por ejemplo, un administrador de red deberá enviar un mensaje en formato XML al servidor, adjuntando una solicitud dentro de un elemento <rpc>. Luego el servidor deberá devolver una respuesta con un elemento <rpc-reply> que contendrá ya sea la información solicitada, un elemento <ok> de operación exitosa o un elemento <rpc-error> notificando algún tipo de error [7]. La sintaxis y la codificación XML de los RPCs de la capa de mensajes están definidos por el esquema XML, ubicado en el Apéndice B del RFC 6241.

1.4.4.1 Elemento <rpc>

Encierra una solicitud de NETCONF enviada desde el cliente al servidor. El elemento <rpc> contiene un atributo obligatorio "message-id", que es una cadena (srting) elegida por el remitente de la RPC, la cual va incrementando en una unidad. El receptor no codifica dicha cadena, sino que la almacena para usarla como atributo del “message-id" en la respuesta <rpc-reply> [7].

1.4.4.2 Elemento <rpc-reply>

Se envía en respuesta a un mensaje <rpc>. Tiene un atributo obligatorio "message-id", que es igual al atributo "message-id" del mensaje <rpc>. Cualquier atributo adicional enviado por el <rpc> se devuelve sin modificar en la respuesta del <rpc-reply> [7].

Un elemento <rpc-replay> puede contener una jerarquización de datos que fue solicitada por parte del cliente, así también puede contener las siguientes estructuras <rpc>:

1. Elemento <rpc-error>: Se envía en el caso de que ocurra un error en el proceso de solicitud de <rpc>. En el caso de existir múltiples errores, el <rpc-reply> contendrá varios elementos <rpc-error> [7].

2. Elemento <ok>: Se envía si no existió errores o alarmas durante el proceso de solicitud de <rpc>. También se envía en el caso de que no se retorne datos de la solicitud realizada [7].

1.4.5 Capa de operaciones

1.4.5.1 Datos de configuración y datos de estado

La información que se puede recuperar de un sistema de ejecución está separada en dos clases, datos de configuración y datos de estado. La primera representa un conjunto de datos de escritura que permiten transformar el estado inicial de un sistema a un estado

deseado. El segundo grupo son los datos adicionales en un sistema que no son datos de configuración, como por ejemplo la información de estado de lectura o las estadísticas recopiladas. El protocolo NETCONF reconoce la diferencia entre datos de configuración y datos de estado y proporciona operaciones para tratar con los mismos [7].

1.4.5.2 Almacenes de datos de configuración

NETCONF define la existencia de uno o más almacenes de datos de configuración en los cuales se puede realizar operaciones de configuración. Un almacén de datos de configuración se define como un conjunto completo de datos de configuración para que un dispositivo de red pase de su estado predeterminado inicial a un estado operativo deseado [7].

El almacén de datos de configuración predeterminado es el almacén <running>, que es aquel que contiene la configuración completa que actualmente está activa en un dispositivo de red. Solo existe un almacén de datos de este tipo en el dispositivo y siempre está presente. La existencia de los otros almacenes de datos de configuración que permite NETCONF depende principalmente de las capacidades soportadas por el dispositivo [7].

Por ejemplo, el almacén de datos de configuración <startup> depende de la capacidad Startup y es aquel que es cargado por el dispositivo como parte de su inicialización cuando se reinicia o se recarga. Las operaciones que afectan a la configuración <running> no se copian automáticamente en la configuración <startup>. Es por eso que es necesario emplear una operación denominada <copy-config> [7].

Otro ejemplo, es el almacén de datos de configuración <candidate>. Este depende de la capacidad Candidate y se usa comúnmente para almacenar los datos de configuración que se pueden manipular sin afectar la configuración actual del dispositivo. Es ideal para construir el modelo de datos de configuración deseado. Se suele emplear la operación <commit> para copiar el contenido del almacén de datos <candidate> al almacén de datos <running> [7].

1.4.5.3 Operaciones Base

NETCONF proporciona un pequeño conjunto de operaciones de bajo nivel para administrar configuraciones de dispositivos y recuperar información de estado de dispositivos. Dichas operaciones son capaces de recuperar, configurar, copiar, y eliminar almacenes de datos de configuración [7]. La Tabla 1.2 basada en [7] , muestra a manera resumida las operaciones base de NETCONF.

Tabla 1.2. Operaciones base de NETCONF

Operación base	Descripción	Parámetros
<get>	Recupera la información de configuración y la información de estado del almacén de datos <running> de un dispositivo.	**filter:** Especifica una parte de los datos de configuración y de estado del dispositivo a recuperar. Si el parámetro no está presente se recupera toda la información de configuración y de estado.
<get-config>	Recupera toda o parte de la información de configuración contenida en un almacén de datos especifico.	**source:** Nombre del almacén de datos de configuración al que se solicita la información. **filter:** Especifica una parte del almacén de datos de configuración del dispositivo a recuperar. Si el parámetro no está presente se recupera toda la configuración es recuperada.
<edit-config>	Carga todo o parte de una configuración a un almacén de datos especifico.	**target:** Nombre del almacén de datos de configuración a editar. **default-operation:** Define valores para establecer la forma de operación de <edit-config>. **test-option**: Define valores para evaluar la existencia de errores al realizar la operación <edit-config>. Solo es disponible si el dispositivo permite la capacidad Validate. **error-option:** Define valores que determinan el funcionamiento de <edit-config> en el caso de la existencia de errores.
<copy-config>	Remplaza un almacén de datos de configuración completo por otro.	**target:** Nombre del almacén de datos de configuración empleado como el destino de la operación <copy-config> **source:** Nombre del almacén de datos de configuración empleado como la fuente de la operación <copy-config>
<delete-config>	Elimina un almacén de datos de configuración.	**target:** Nombre del almacén de datos de configuración a eliminar.
<lock>	Bloquea un almacén de datos de configuración. dando así seguridad al cliente de poder realizar configuraciones sin que otros clientes intervengan.	**target:** Nombre del almacén de datos de configuración a bloquear.

<unlock>	Desbloquea un almacén de datos de configuración.	**target:** Nombre del almacén de datos de configuración a desbloquear.
<close-session>	Cierra la sesión de NETCONF correctamente	-
<kill-session>	Fuerza el cierre de una sesión NETCONF	**session-id:** Es el identificador de sesión NETCONF que va a ser forzada a terminar.

Las operaciones son solicitudes que son encapsuladas en mensajes <rpc> y son enviadas al servidor. Las respuestas a dichas solicitudes se devuelven en mensajes <rpc-reply>. En el ANEXO I se muestran ejemplos de dichos mensajes al emplear las operaciones mostradas en la Tabla 1.2.

1.4.5.4 Capacidades NETCONF

Una capacidad de NETCONF es un conjunto de funciones extra que aumentan las especificaciones base del protocolo. Las capacidades ofrecen nuevas operaciones, así también permiten mejorar el rendimiento de las operaciones base. Un cliente puede hacer uso de las capacidades de un servidor, si estas fueron anunciadas por el mismo en el intercambio inicial de mensajes <hello> que se dan durante el inicio de sesión. El cliente será capaz de usar cualquier operación, parámetro y contenido adicional definido por esas capacidades [7]. Las capacidades de NETCONF se identifican con un URI y están definidas en el RFC 6241 con el siguiente formato:

urn:ietf:params:netconf:capability:{name}:{x.x}

donde “name” representa el nombre de la capacidad y “x.x” representa la versión de la capacidad [7].

En el ANEXO II se describen algunas capacidades NETCONF con sus respectivos ejemplos.

1.4.6 Capa de contenido: Lenguaje de modelado de datos YANG

YANG es un lenguaje estructurado y fuertemente tipado para describir modelos de datos [4], [10]. Diseñado específicamente para modelar datos de configuración, datos de estado, RPCs y notificaciones del protocolo de gestión de red NETCONF [10]. Aunque, YANG también puede ser usado por otros protocolos como por ejemplo RESTCONF y CoAP [12].

YANG, estructura los modelos de datos en módulos y submódulos [10]. A un modelo de datos individual se lo conoce como “Módulo YANG” [4], y es aquel que define una jerarquía de datos que es utilizada para las operaciones base, RPCs y notificaciones de NETCONF,

permitiendo así una descripción completa de todos los datos enviados entre cliente y servidor [10].

YANG modela la organización jerárquica de los datos como un árbol en el que cada nodo tiene un nombre y un valor o un conjunto de nodos secundarios. YANG proporciona una descripción concisa y clara de cada nodo, así como la relación entre esos nodos [10].

Un módulo YANG es capaz de importar datos de otros módulos externos, así como incluir datos de submódulos. También existe la posibilidad de aumentar el contenido de la jerarquía de un módulo, ya que se puede adicionar otros nodos de datos que fueron definidos en otro módulo [10].

1.4.6.1 Módulos y Submódulos

Un módulo está compuesto de 3 tipos de declaraciones. La primera declaración es el encabezado que describe al módulo y ofrece información adicional sobre él. La segunda es la declaración de revisión que brinda información sobre la historia del módulo. Por último, la declaración de definición que es el cuerpo del módulo donde el modelo de datos se define [10].

Un servidor NETCONF puede emplear uno o varios módulos para representar datos. Tener varios módulos permite tener múltiples vistas de los mismos datos o múltiples vistas de las subsecciones de los datos del dispositivo [10]. Por ejemplo, un servidor puede tener varios módulos para representar los datos de las interfaces que dispone.

Un módulo puede contener módulos y submódulos externos. Se suele emplear la sentencia “import” para incorporar nodos o datos de un módulo externo, mientras la sentencia “include” se emplea para hacer referencia al contenido de un submódulo [10].

1.4.6.2 Tipos de nodos

Se define 4 principales nodos de datos empleados en el modelado de datos YANG [10], los cuales se detallan en la Tabla 1.3.

Tabla 1.3. Principales nodos de datos en YANG

Nodos	Descripción
leaf	Se usan para representar un único atributo/dato.
leaf-list	Es una secuencia de nodos leaf, donde cada uno posee un único atributo/dato del mismo tipo.
container	Se utiliza para agrupar nodos relacionados.
list	Permite crear una estructura de nodos, específicamente una secuencia de n listas, las cuales son identificas únicamente por los valores de sus hojas.

El ANEXO III contiene ejemplos que muestran la sintaxis YANG con su respectiva codificación XML de los nodos de la Tabla 1.3.

1.4.7 Proveedores que permiten NETCONF

NETCONF apunta a ser uno de los protocolos de gestión de red más empleados por los administradores de red. Es por esta razón que es necesario definir su situación actual en el ámbito empresarial. Se ha escogido analizar tres de los mayores fabricantes de routers y switches, con el objeto de descubrir aquellos dispositivos que soportan el protocolo NETCONF.

Los fabricantes escogidos fueron Cisco, Juniper y Huawei. El ANEXO IV contiene una lista de los dispositivos que implementan NETCONF para cada fabricante.

2 METODOLOGÍA

Para el desarrollo de este trabajo de integración curricular se siguió la metodología detallada en las siguientes etapas:

1. **Investigación documental**: Recopilación y selección de información a través de consultas de referencias bibliográficas como textos, artículos, estándares, así como fuentes informáticas confiables.
2. **Selección de herramientas:** Pruebas de distintas herramientas de simulación o emulación, selección de esquema de red a utilizar en las prácticas y la elección de dispositivos de red a emplear.
3. **Selección de temas y objetivos**: Segmentación de la información y planteamiento de temas y objetivos a abordar en las prácticas de laboratorio.
4. **Desarrollo del contenido de las prácticas**: Generación del procedimiento a seguir en cada una de las prácticas. Se desarrollaron enunciados, preguntas y ejemplos de aplicación.
5. **Diseño de las prácticas**: Generación de la estructura de las prácticas de laboratorio. Se creó cada uno de los ítems establecidos en el formato de prácticas de laboratorio del Departamento de Electrónica, Telecomunicaciones y Redes de Información de la Escuela Politécnica Nacional.
6. **Análisis de resultados**: Pruebas de funcionamiento del procedimiento de cada una de las prácticas y resolución de informes.

2.1 Selección de herramientas

La Figura 2.1 muestra el esquema de red que se ha propuesto emplear en todas las prácticas de laboratorio. En la misma se puede apreciar dos dispositivos. Una máquina virtual que hace la función de cliente NETCONF y un dispositivo de red que hace la función del servidor NETCONF. Debido a que el propósito de las prácticas es abordar de la manera más sencilla los fundamentos teóricos de NETCONF, la topología de red propuesta es suficiente para cumplir con este requerimiento.

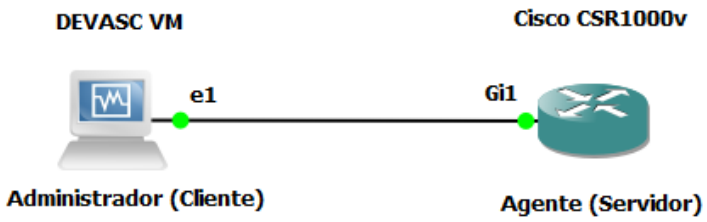


Figura 2.1 Esquema de red propuesto

Para el desarrollo de las prácticas de laboratorio se ha empleado el emulador GNS3. La principal razón de su uso es que permite emular el hardware de una variedad de routers y switches [13]. En otras palabras, admite ejecutar un archivo IOS real en el computador, logrando que todos los comandos de configuración y salidas provengan de una IOS real. Teóricamente todos los protocolos y características disponibles en la versión del IOS estarán disponibles para su uso [13]. Debido a que NETCONF es soportado solo en algunos dispositivos de red, la tarea fue buscar aquellos en la lista de dispositivos, también denominados “Appliances”, que son permitidos en GNS3. De la lista se escogió el router Cisco CSR1000v como el dispositivo de red para la realización de las prácticas de laboratorio. Específicamente se seleccionó emplear Cisco IOS XE 17.3.2 como la versión del software del dispositivo.

Hacer uso del dispositivo involucró ciertas consideraciones. Primero, se requirió buscar un archivo de imagen de disco en formato QEMU (Quick Emulator), esto debido a que GNS3 emplea el emulador QEMU para poder arrancar el sistema operativo del router Cisco CSR1000v. Comúnmente, los archivos imágenes de disco de QEMU poseen la actual extensión .qcow2, que representa la segunda versión de QEMU copy-on-write. Segundo, fue necesario buscar los requisitos de instalación de dicho archivo. Según [14] para poder instalar Cisco IOS XE 17.3.2 en entornos Linux/KVM (Kernel-based Virtual Machine), se necesita de una máquina virtual y uso de archivos del tipo .iso o .qcow2. Además, se menciona que los requerimientos mínimos de instalación son 1vCPU con al menos 4GB RAM y 8 GB de tamaño de disco duro virtual. Por último, fue necesario integrar GNS3 con

la máquina virtual GNS3 (GNS3 VM) con el propósito de que esta última ejecute las imágenes, contenedores y máquinas virtuales. En otras palabras, se estableció la máquina virtual GNS3 como el servidor para ejecutar la imagen del router. Es por esto que se configuró la misma teniendo en cuenta los requisitos mínimos de instalación de Cisco IOS XE 17.3.2 para así evitar que este último presente inconvenientes al momento de ser utilizado.

La segunda razón del uso de GNS3 es que permite integrar sistemas operativos emulados, logrando conectar los mismos en red con otros dispositivos de GNS3 [13]. Previamente se explicó que se decidió usar el router Cisco CSR1000v como el dispositivo de red que cumpla la función de servidor NETCONF para las prácticas de laboratorio. Sin embargo, hace falta la integración de un dispositivo que cumpla el papel de cliente NETCONF. Se ha decidido emplear la máquina virtual DEVASC de Cisco debido a que brinda herramientas que resultan ser útiles en las prácticas de laboratorio. Según Cisco el único requerimiento para usar DEVASC es que el computador host posea al menos 4 GB de RAM y 15 GB de espacio libre en disco duro.

2.2 Temas de las prácticas de laboratorio

Las prácticas de laboratorio fueron segmentadas considerando los siguientes parámetros: el contenido a tratar en las prácticas que debe ser secuencial y no debe abordar temáticas extensas, la dificultad para realizar las configuraciones y/o actividades planteadas y el tiempo estimado para realizar la práctica y analizar los resultados. Teniendo en cuenta estas consideraciones se plantearon cinco prácticas, las cuales se detallan a continuación.

2.2.1 Práctica 1: Introducción a NETCONF

La primera práctica posee dos objetivos que son: implementar el entorno de trabajo en GNS3 con el objeto de emular un entorno adecuado para el uso del protocolo NETCONF y realizar una conexión NETCONF entre cliente y servidor.

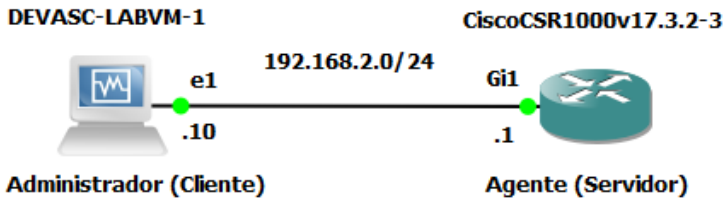


Figura 2.2 Topología de red

La instalación de las herramientas del entorno de trabajo, el armado de la topología de red mostrada en la Figura 2.2 y la configuración de los elementos de dicha topología para

establecer conectividad entre los dispositivos son las actividades que aborda la primera sección de la práctica para satisfacer el primer objetivo. Es primordial guiar a los estudiantes en toda la implementación del entorno de trabajo ya que el mismo será utilizado en las posteriores prácticas de laboratorio. Es por esta razón que se desarrolló una serie de pasos detallados que permitirán a los estudiantes implementar de la mejor manera el entorno de trabajo.

En virtud de cumplir el segundo objetivo de la práctica, se desarrolló una serie de pasos a seguir una vez culminadas las actividades anteriormente mencionadas. La configuración de SSH y la habilitación del servicio NETCONF en el router Cisco CSR1000v son las actividades a efectuar por los estudiantes para lograr establecer una sesión NETCONF entre cliente y servidor.

Una sesión NETCONF se establece al finalizar el intercambio de mensajes <hello> entre cliente y servidor. La práctica contiene pasos detallados para conseguir establecer dicha sesión, así también describe las salidas obtenidas al seguir los pasos. Es importante destacar que el estudiante tendrá como tarea final verificar el establecimiento de sesión NETCONF y sacar sus propias conclusiones al terminar la práctica.

2.2.2 Práctica 2: Capa de transporte seguro de NETCONF

La segunda práctica aborda el tema de la capa segura de transporte de NETCONF. A pesar de que NETCONF no define un protocolo de transporte seguro especifico a utilizar, SSH se ha designado como el protocolo obligatorio para cualquier implementación de NETCONF. El principal objetivo de esta segunda práctica es lograr que los estudiantes ejecuten el protocolo NETCONF dentro de una sesión SSH empleando dos métodos de autenticación de usuario: por contraseña y por clave pública. Debido a que el protocolo de transporte escogido es el único responsable de autenticar y ofrecer seguridad a las sesiones NETCONF. La práctica desarrollada muestra como SSH hace uso de distintos métodos de autenticación de usuario para poder ofrecer un cliente autenticado, denominado usuario NETCONF, a las otras capas de NETCONF.

Los estudiantes dispondrán en primer lugar de una serie de pasos para configurar la autenticación de usuario de SSH por contraseña al emplear una base de datos de usuarios local y una remota, siendo esta última implementada por un servidor de usuarios remoto TACACS+ (Ver Figura 2.3). Es importante mencionar que se empleó AAA para la autenticación de usuario que requiere SSH. Una de las características de NETCONF es que sigue de cerca la funcionalidad nativa de los dispositivos, por lo que SSH no tendrá inconvenientes de emplear los métodos de autenticación disponibles en el router Cisco

CSR1000v para autenticar usuarios NETCONF. Uno de estos métodos es AAA, que es la función nativa del router utilizada para autenticar, autorizar y registrar la actividad de los usuarios.

Figura 2.3 Topología de red con servidor TACACS+

Luego los estudiantes dispondrán de una serie de pasos para configurar la autenticación de usuario de SSH por medio del método de clave pública. Para esta parte, los estudiantes deberán volver a emplear la topología de red mostrada en la Figura 2.2.

El estudiante tendrá como tarea final verificar cada una de las configuraciones y sacar sus propias conclusiones. En posteriores prácticas el estudiante podría emplear cualquier método de autenticación de usuario para SSH visto en esta práctica, sin embargo, es importante mencionar que el método que será utilizado en las siguientes prácticas será el de clave pública.

2.2.3 Práctica 3: Lenguaje de modelado de datos YANG

La tercera práctica corresponde a la capa de contenido del protocolo NETCONF, específicamente aborda el tema del lenguaje de modelado de datos YANG. Es recomendable que antes de abordar la capa de mensajes y operaciones de NETCONF se explique y se comprenda la función de YANG. Esto permitirá que los estudiantes no tengan problemas y sobre todo tengan mayor facilidad de entendimiento al tratar con mensajes y operaciones NETCONF.

El objetivo de la tercera práctica es lograr que los estudiantes comprendan que es el leguaje YANG y que son los modelos de datos YANG. Con esto aprenderán que NETCONF emplea en lenguaje YANG para poder crear modelos de datos, los cuales pueden ser empleados en los dispositivos de red para representar sus datos.

En primer lugar, la práctica se enfoca en explorar los modelos de datos YANG. Para ello se utilizó un repositorio de GITHUB el cual contiene una colección de módulos YANG. Los estudiantes tendrán que acceder a dicho repositorio, seleccionar los modelos de datos de CISCO para la versión IOS XE 17.3.1 y analizar el contenido de los modelos ietf-interfaces y openconfig-interfaces.

Luego los estudiantes emplearán pyang para poder convertir los modelos YANG explorados anteriormente en formato de árbol, logrando así que los modelos sean mucho más fáciles de entender en su totalidad. En esta parte los estudiantes tendrán que analizar cada una de las transformaciones.

Las últimas actividades de esta práctica permitirán que los estudiantes entiendan como los modelos de datos YANG son empleados para representar los datos de un dispositivo de red. Los estudiantes tendrán que analizar un ejemplo de los datos devueltos desde un dispositivo de red que muestra información sobre sus interfaces y luego compararlo con el modelo de datos ietf-iterfaces. Por último, los estudiantes deberán identificar el modelo de datos ietf-iterfaces en la lista de capacidades del router Cisco CSR1000v, para luego mediante una operación <get> solicitar la información de las interfaces de acuerdo al modelo de datos identificado.

El estudiante tendrá como tarea final responder a las preguntas y/o enunciados presentados en el desarrollo de la práctica y sacar sus propias conclusiones.

2.2.4 Práctica 4: Operaciones y capacidades NETCONF

La cuarta práctica aborda la capa de mensajes y la capa de operaciones NETCONF. Específicamente se enfoca en el modelo RPC, operaciones base y capacidades. Los objetivos de esta práctica son comprender como los mensajes <rpc> y <rpc-reply> son empleados para el intercambio de información, explicar el funcionamiento de las operaciones base y comprender que es una capacidad NETCONF.

La primera actividad de la práctica consiste en explorar las capacidades del router Cisco CSR1000v. Los enunciados propuestos permitirán identificar entre una capacidad NETCONF y un modelo de datos anunciado como capacidad, así también permitirán explicar las capacidades NETCONF y los almacenes de datos de configuración soportado por el router Cisco CSR1000v.

Las operaciones de NETCONF son solicitudes que son encapsuladas en mensajes <rpc> y son enviadas al servidor. Que luego son respondidas con mensajes <rpc-reply>. El estudiante tendrá como segunda actividad enviar mensajes <rpc> con las operaciones base expuestas en la Tabla 1.2. La práctica fue diseñada con el objeto de facilitar a los estudiantes el uso de las operaciones, es por esta razón que cada enunciado contiene el texto en XML que deberán ser enviado al servidor, así también se ofrece una explicación de dicho texto.

Como última actividad de la práctica, los estudiantes verificarán el funcionamiento de algunas capacidades NETCONF anunciadas por el router Cisco CSR1000v. Los enunciados de esta parte seguirán la misma metodología que los enunciados de las operaciones.

El estudiante tendrá como tarea final analizar las respuestas enviadas por parte del servidor luego de realizar una solicitud, responder las preguntas y/o enunciados presentados en el desarrollo de la práctica y sacar sus propias conclusiones.

2.2.5 Práctica 5: Cliente NETCONF con Python

NETCONF ofrece una formal y completa API, lo que permite el uso de la programación para gestionar las redes. El propósito de la quinta práctica es emplear Python, específicamente la librería ncclient, para poder enviar información hacia el router Cisco CSR1000v, así como recibir información del mismo.

Para el desarrollo de toda la práctica se empleó el software Visual Studio Code (VSC) de la máquina virtual DEVASC, esto debido a que facilita la programación con Python. En esta práctica los enunciados disponen del código y una explicación del mismo. Esto con el propósito de facilitar a los estudiantes el uso de Python.

Como primera actividad los estudiantes emplearán los códigos propuestos para establecer una sesión NETCONF entre cliente y servidor. Dichos códigos permitirán el uso de una autenticación por contraseña y el uso de una autenticación por clave pública.

El resto de actividades se enfocará en la recuperación y envió de información. Los estudiantes harán uso de un código para recuperar información de las interfaces del dispositivo, ocuparán otros para editar la información del router Cisco CSR1000v y por último emplearán uno para guardar la información configurada en la configuración de inicio del dispositivo.

El estudiante tendrá como tarea final analizar cada uno de los códigos empleados, responder las preguntas y/o enunciados presentados en el desarrollo de la práctica y sacar sus propias conclusiones.

2.3 Estructura de las prácticas de laboratorio

Las prácticas diseñadas fueron escritas siguiendo la estructura de prácticas de laboratorio establecido por el Departamento de Electrónica, Telecomunicaciones y Redes de Información de la Escuela Politécnica Nacional. Cada una de las prácticas diseñadas poseen las ocho partes establecidas en el formato, las cuales son: Tema, Objetivos, Marco teórico, Trabajo preparatorio, Equipos y materiales, Procedimiento, Informe y Referencias.

1. **Tema:** Contiene el nombre que identifica a la práctica de laboratorio. Ofrece una idea general del contenido que se abordará en las prácticas.
2. **Objetivos:** Esta parte contiene un total de tres objetivos, los cuales representan las metas que se alcanzarán luego de desarrollar las diferentes actividades propuestas en las prácticas de laboratorio.
3. **Marco Teórico:** Muestra de forma sintetizada la teoría relaciona a la temática de las prácticas de laboratorio. Su función es proveer información simple y de fácil abstracción que será de utilidad para comprender de mejor manera las actividades que serán ejecutadas en el desarrollo de las prácticas.
4. **Trabajo preparatorio:** Son actividades previas a realizar las prácticas de laboratorio. En esta parte se solicita a los estudiantes descargar algún recurso o consultar sobre alguna temática relacionada con la práctica.
5. **Equipos y materiales:** Contiene los equipos o materiales que son necesarios para el desarrollo de la práctica. Un computador con el software GNS3 es lo único que se requiere en cada una de las prácticas.
6. **Procedimiento:** Es la parte más representativa de las prácticas de laboratorio. En esta se detalla el proceso a seguir por los estudiantes para alcanzar con éxito los objetivos de cada práctica. Se propusieron diversas actividades con su debida explicación; como realizar alguna configuración, analizar y contestar algunas preguntas, recuperar o solicitar información e incluso crear algún programa.
7. **Informes:** Contienen actividades que pueden ser teóricas o prácticas que deben realizar los estudiantes. Dichas actividades permiten evaluar los conocimientos adquiridos en las prácticas de laboratorio, así como reforzar los mismos.
8. **Referencias:** Contiene las fuentes bibliográficas empleadas para el desarrollo de las prácticas de laboratorio.

2.4 Prácticas de laboratorio

El conjunto de prácticas de laboratorio desarrollado como componente de este trabajo de titulación se muestran en esta sección. Debido a cuestiones de espacio solo se mostrará la hoja guía de la práctica 1, el resto de prácticas se encontrarán en el ANEXO V.

2.4.1 Práctica 1

2.4.1.1 Tema

Introducción a NETCONF

2.4.1.2 Objetivos

- Implementar el entorno de trabajo en GNS3 para el uso del protocolo NETCONF.
- Configurar el router cisco Cloud Services Router 1000v para habilitar el servicio NETCONF.
- Realizar una conexión NETCONF entre Administrador (Cliente) y Agente (Servidor).

2.4.1.3 Marco teórico

¿Qué es NETCONF?

NETCONF es un protocolo de gestión de red que define un mecanismo para la administración de dispositivos de red con el que se puede recuperar tanto información de configuración como de estado, así como cargar y manipular datos de configuración [7]. Entre las características de NETCONF se mencionan las siguientes [7]

- Ofrece a los dispositivos una completa y formal API (Application Programming Interface), permitiendo así el uso de aplicaciones o scripts que son empleados para la administración de los dispositivos.
- Obliga el uso de protocolos de transporte seguros como por ejemplo SSH.
- Realiza una distinción entre información de estado y de configuración.
- Brinda una estructura de respuestas de errores.
- Sigue de cerca la funcionalidad nativa de los dispositivos, permitiendo que las aplicaciones accedan al contenido sintáctico y sistemático de las interfaces de los usuarios.
- Permite que los clientes descubran las funcionalidades que son soportadas por el servidor, a las que se las denomina “Capacidades”. Con estas el cliente es capaz de mejorar el rendimiento de la administración de los dispositivos.

Conceptualmente NETCONF se divide en cuatro capas [7] (ver Figura 2.4), las cuales se describen en la Tabla 2.1.

Tabla 2.1. Descripción de las capas de NETCONF

1	Capa de Transporte Seguro	Proporciona una ruta de comunicación segura entre cliente y servidor.
2	Capa de Mensajes	Facilita un mecanismo de estructuración de tramas simples para codificar RPC y notificaciones.
3	Capa de Operaciones	Establece un conjunto de operaciones base del protocolo NETCONF.

4	Capa de Contenido	No definida en el estándar RFC 6241. Sin embargo, es cubierto por el lenguaje de modelado de datos YANG ya que especifica los modelos de datos NETCONF y las operaciones del protocolo de la capa 3.

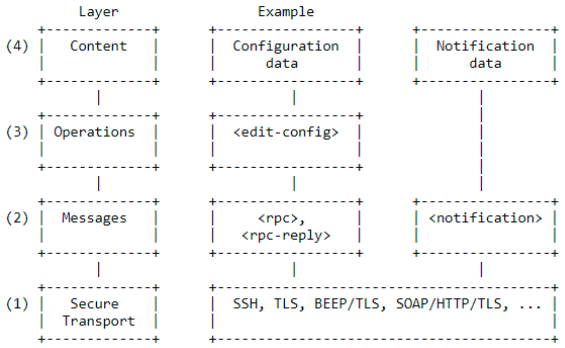


Figura 2.4. Estructura de capas de NETCONF [7]

NETCONF en el contexto actual

NETCONF ha sido adoptada rápidamente por muchos proveedores y clientes debido a que se presenta como una solución a los inconvenientes del protocolo SNMP y la CLI para la administración de redes. Actualmente este protocolo continúa evolucionando y ganando mayor aceptación [4]. La situación actual de NETCONF en el ámbito empresarial está tomando fuerza, esto debido a que muchos fabricantes de routers y switches como por ejemplo CISCO, JUNIPER y HUAWEI ya ofrecen algunos dispositivos que soportan el protocolo NETCONF.

2.4.1.4 Trabajo Preparatorio

a. Descargar la imagen de disco del router Cisco Cloud Services Router 1000v versión 17.3.2 en formato QEMU (*csr1000v-universalk9.17.03.02-serial.qcow2*) del siguiente link: https://drive.google.com/drive/folders/1PpffXFFhZ53bHO47ZprtouNtB7o1oelz

b. Descargar el archivo OVA de la máquina virtual DEVASC-LABVM del siguiente link:https://www.netacad.com/portal/content/devnet-associate-virtual-machines-vms

   NOTA: Es necesario tener una cuenta en DEVNET.

c. Descargar e instalar GNS3. Puede utilizar el siguiente link de descarga: https://www.gns3.com/software/download

d. Descargar el archivo OVA de la máquina virtual GNS3 (VMware Workstation and Fusion) del siguiente link: https://www.gns3.com/software/download-vm

e. Descargar los siguientes softwares de virtualización:

VirtualBox: https://www.virtualbox.org/wiki/Downloads

VMware Workstation Player: https://www.vmware.com/latam/products/workstation-player/workstation-player-evaluation.html

f. Consultar los requisitos mínimos de vCPU, tamaño de memoria RAM y disco duro virtual para la instalación de la imagen de disco del router CISCO CSR1000v versión IOS XE 17.3.2 en entornos KVM.

g. Consultar y describir 3 dispositivos (routers o switches) que soporten el protocolo NETCONF.

h. Consultar sobre la estructura de red básica de NETCONF.

2.4.1.5 Equipos y materiales

- PC (GNS3)

2.4.1.6 Procedimiento

Actividad 1: Implementación del entorno de trabajo

a. Instalar máquina virtual GNS3.

- Extraer el archivo .zip que fue descargado como parte del preparatorio para obtener el archivo *GNS3 VM.ova*.

- Abrir VMware Workstation Player, seleccionar “Abrir una Máquina Virtual” y escoger el archivo *GNS3 VM.ova*.

- Configurar el nombre de la máquina virtual y hacer clic en “Importar”. (ver Figura 2.5)

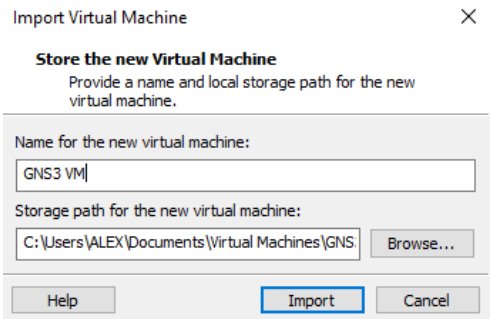


Figura 2.5. Configuración del nombre de la máquina virtual

- Añadir un nuevo adaptador de red y configurarlo en modo Puente. (ver Figura 2.6)

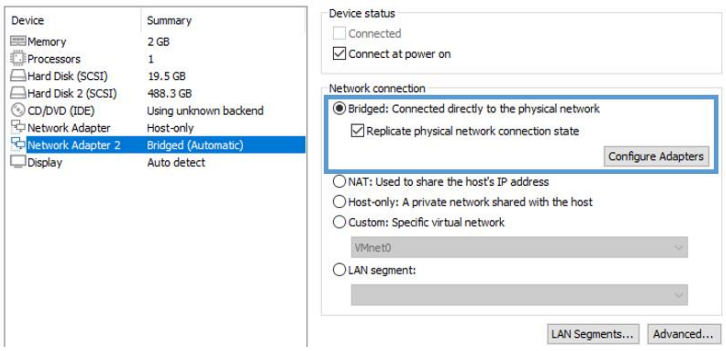


Figura 2.6. Configuración del Adaptador de red en modo Puente

NOTA: Verificar que esté seleccionado el adaptador de red de su computador al utilizar el modo puente. Para ello seleccionar el botón “Configuración de adaptadores” que se muestra en la Figura 2.6.

b. Integrar GNS3 con la máquina virtual GNS3 (GNS3 VM) [15].

- Seleccionar “Ayuda” y dar clic en el “Asistente de configuración”. (ver Figura 2.7)

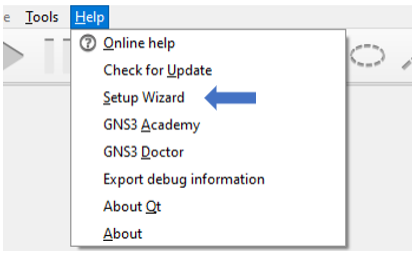


Figura 2.7. “Asistente de configuración” en menú “Ayuda”

- Escoger la opción “Ejecutar los dispositivos en una máquina virtual”. (ver Figura 2.8)

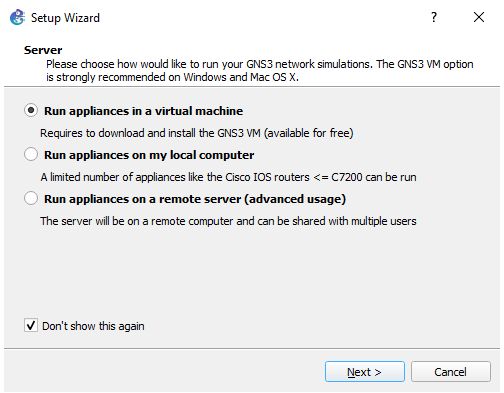


Figura 2.8. Ventana para escoger el servidor de ejecución de los dispositivos

- Aunque GNS3 VM ejecutará las máquinas virtuales/imágenes/contenedores, debe configurar el servidor local con los parámetros que se muestran en la Figura 2.9.

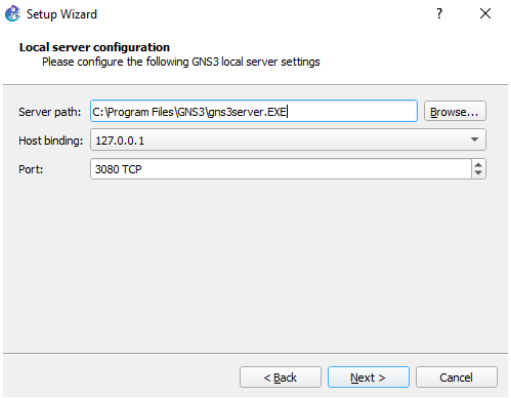


Figura 2.9. Ventana para configurar el servidor local

- Validar la correcta conexión del servidor local (ver Figura 2.10) y seleccionar “Siguiente”.

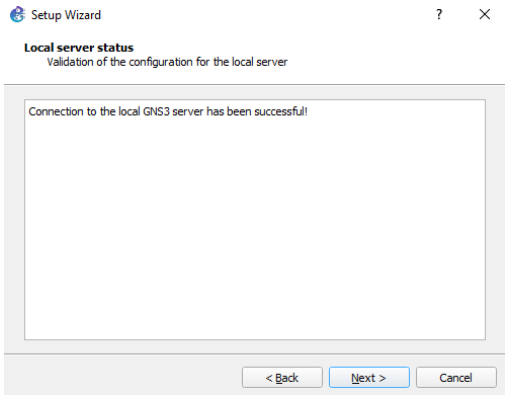


Figura 2.10. Ventana para validar la conexión con el servidor local

- Presionar el botón “Refrescar” en el caso de que no se detecte automáticamente la presencia de la máquina virtual GNS3 de WMware Workstation. Si no es el caso configurar el número de vCPUs y el tamaño de la RAM con los parámetros mostrados en la Figura 2.11.

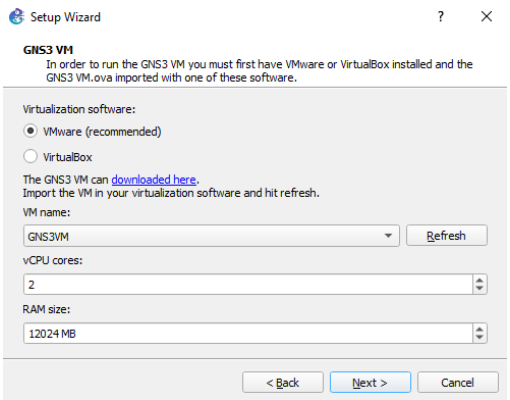


Figura 2.11. Configuración de parámetros de GNS3 VM

- Se mostrará un resumen de la configuración de la máquina virtual GNS3. Dar clic en "Finalizar". (ver Figura 2.12)

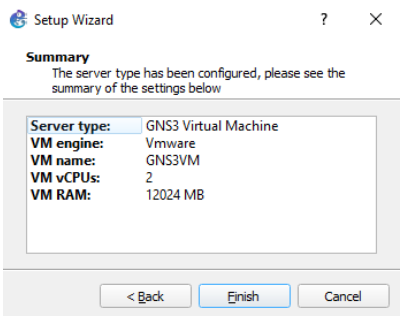


Figura 2.12. Resumen de la configuración de GNS3 VM

- De acuerdo a lo consultado en el trabajo preparatorio, ¿La configuración de la máquina virtual GNS3 fue la adecuada para poder ejecutar la imagen de disco router Cisco CSR1000v? Argumente su respuesta.

c. Importar router Cisco CSR1000v en GNS3.

- Abrir GNS3 y generar una "Nueva plantilla".

NOTA: Una forma sencilla de hacerlo es seleccionar cualquiera de los cuatro iconos de izquierda mostrados en la Figura 2.13 y presionar "Nueva plantilla".

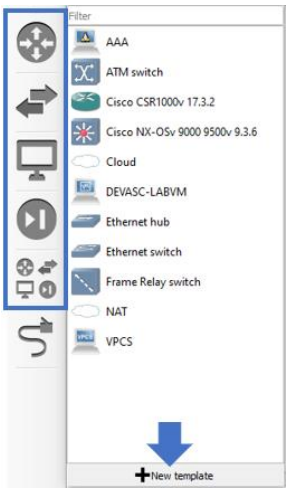


Figura 2.13. Creación de una nueva plantilla

- Seleccionar la opción "Instalar un dispositivo desde el servidor de GNS3". (ver Figura 2.14)

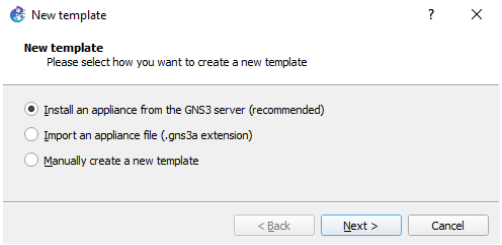


Figura 2.14. Ventana para escoger como crear una nueva plantilla.

- Seleccionar el router Cisco CSR1000v y presionar “Instalar”. (ver Figura 2.15)

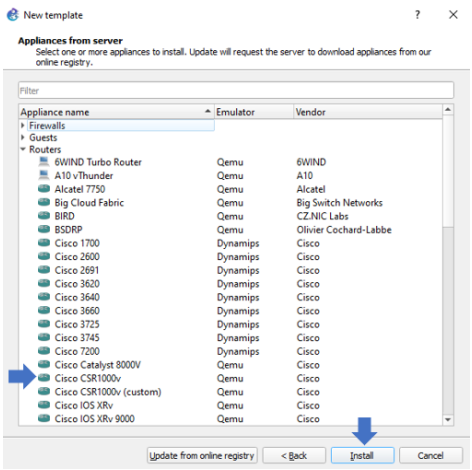


Figura 2.15. Ventana para escoger el router Cisco CSR1000v

- Seleccionar la opción “Instalar el dispositivo en GNS3 VM”. (ver Figura 2.16)

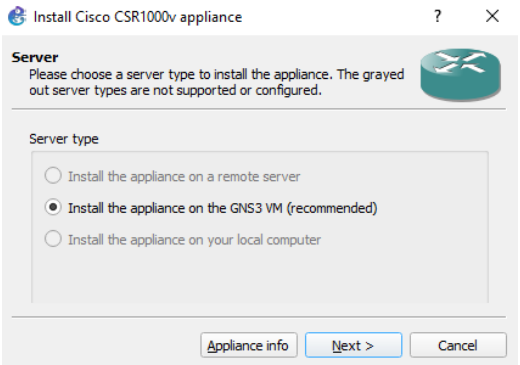


Figura 2.16. Ventana para escoger donde ejecutar el router Cisco CSR1000v

- Dejar por defecto el Qemu binary que será usado por el dispositivo. (ver Figura 2.17)

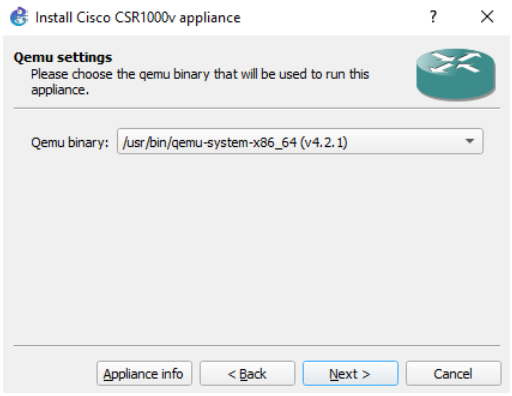


Figura 2.17. Ventana para escoger el Qemu binary

- Debido a que la versión 17.3.2 del router Cisco CSR1000v no se encuentra en la lista, debe crear una nueva versión. (ver Figura 2.18)

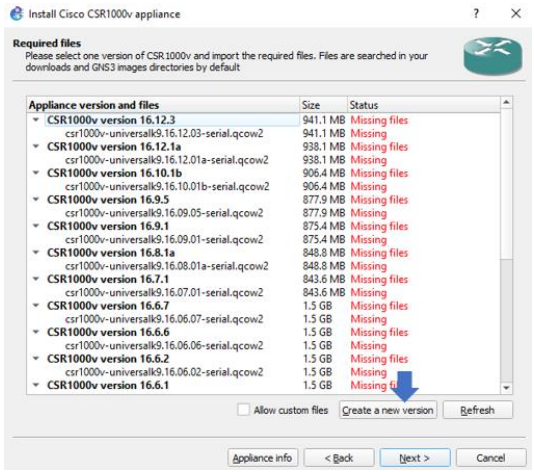


Figura 2.18. Botón “Crear una nueva versión”

- Digitar 17.3.2 como el nombre de la nueva versión. (ver Figura 2.19)

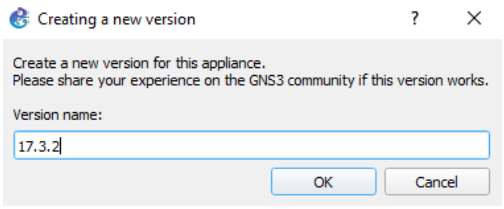


Figura 2.19. Configuración del nombre de la nueva versión

- Colocar el nombre y la respectiva extensión de imagen de disco del router Cisco CSR1000v. (ver Figura 2.20) Recuerde que dicho archivo fue descargado como parte del preparatorio.

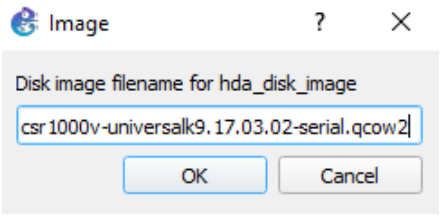


Figura 2.20. Nombre y extensión de la imagen de disco del router Cisco CSR1000v

- Importar el archivo descargado y verificar que esté listo para instalar. Dar clic en "Siguiente". (ver Figura 2.21)

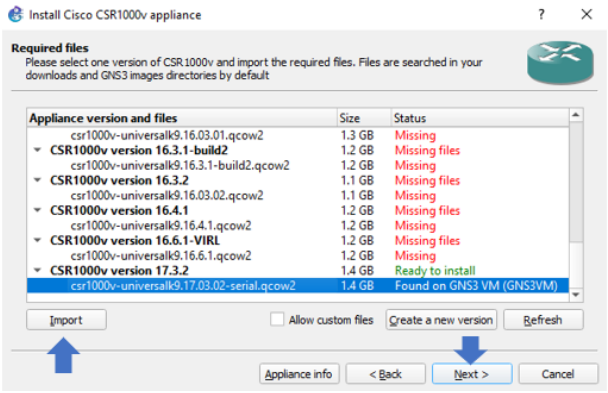


Figura 2.21. Ventana que verifica que la imagen de disco está lista para instalar.

- Si la instalación fue un éxito se mostrará un mensaje que muestre la forma de uso del dispositivo. Dar clic en "Finalizar". (ver Figura 2.22)

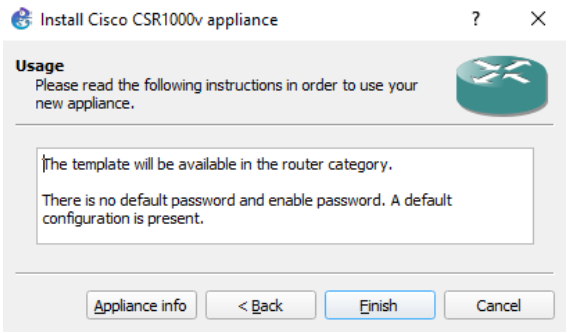


Figura 2.22. Instrucciones de uso de la imagen de disco instalada

d. Instalar la máquina virtual DEVASC [16].

- Abrir VirtualBox, dirigirse a Archivo → Importar servicio virtualizado y seleccionar el archivo OVA de la máquina virtual DEVASC-LABVM. El proceso de importación suelte tardar unos cuantos minutos.

- Configurar el adaptador de red 1 en modo “Adaptador puente” y configurar el adaptador de red 2 en modo “No conectado”. (ver Figura 2.23 y Figura 2.24)

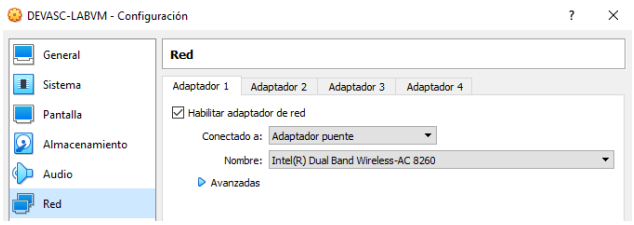


Figura 2.23. Configuración de adaptador de red 1 de DEVASC-LABVM

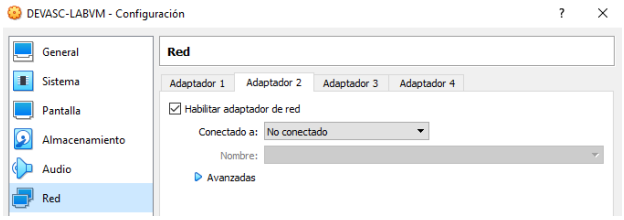


Figura 2.24. Configuración de adaptador de red 2 de DEVASC-LABVM

- Ejecutar la máquina virtual DEVASC-LABVM. Esto tomará algunos minutos mientras se inicia la imagen de Ubuntu.

- Debido a que DEVASC-LABVM posee Packet Tracer, debe aceptar el EULA de Cisco Packet Tracer para continuar con el inicio de máquina virtual. Cuando visualice el acuerdo de licencia, utilice las teclas de fecha para desplazarse por el texto. Seleccione <ok> y presione la barra espaciadora, por último, seleccione <I Agree>. (ver Figura 2.25)

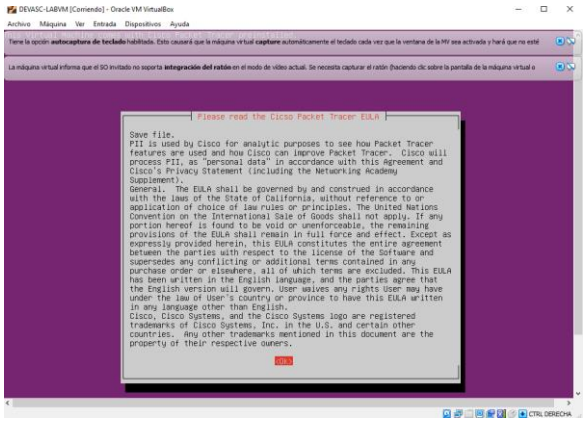


Figura 2.25. EULA de Cisco Packet Tracer.

- La imagen de Ubuntu se procederá a cargar.

e. Integrar máquina virtual DEVASC con GNS3.

- Dirigirse a Edición → Preferencias. (ver Figura 2.26)

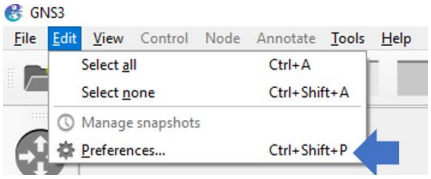


Figura 2.26. “Preferencias” en menú “Edición”

- Seleccionar VirtualBox VMs y dar clic en “Nuevo”.

- Escoger la opción “Ejecutar la máquina virtual de VirtualBox en mi computadora local”. (ver Figura 2.27)

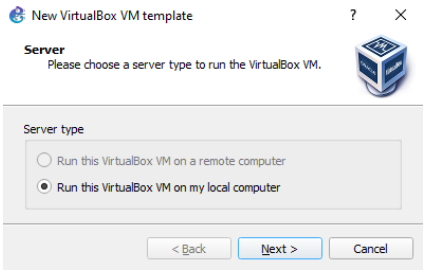


Figura 2.27. Ventana para seleccionar el lugar de ejecución de DEVASC

- Seleccionar la máquina virtual DEVASC-LABVM y dar clic en “Finalizar”. (ver Figura 2.28)

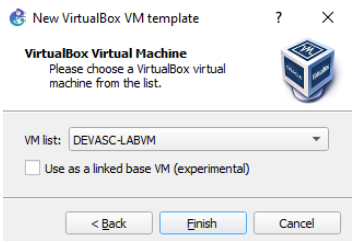


Figura 2.28. Ventana para escoger la máquina virtual DEVASC-LABVM

- Si la máquina virtual fue integrada con éxito en GNS3, aparecerá como una plantilla. (ver Figura 2.29)

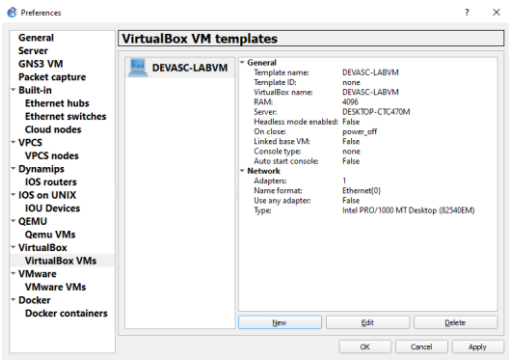


Figura 2.29. Plantilla de DEVASC-LABVM

Actividad 2: Armado de topología de red y establecimiento de conexión entre dispositivos.

a. Abrir GNS3 y conectar la interfaz Ethernet 1 de la máquina virtual DEVASC-LABVM con la interfaz Gigabit Ethernet 1 del router cisco CSR1000v. (ver Figura 2.31)

b. **NOTA:** Una vez arrastrada la máquina virtual DEVASC-LABVM a la ventana central de GNS3, se debe configurar la plantilla con 2 adaptadores de red. (ver Figura 2.30)

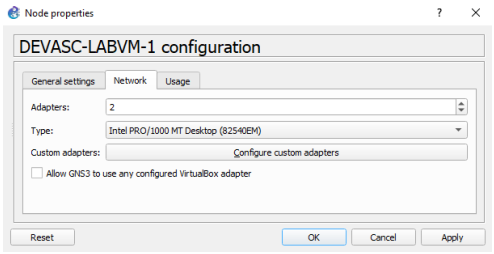


Figura 2.30. Configuración de adaptadores de red en GNS3

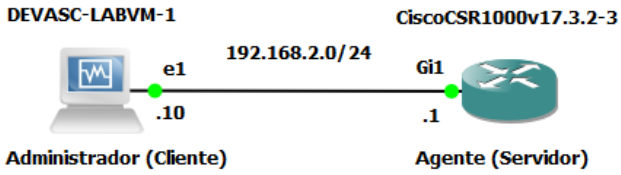


Figura 2.31. Topología de red

c. Verificar el estado de las interfaces de red de la máquina virtual DEVASC-LABVM.

```
devasc@labvm:~$ ifconfig
```

d. Configurar la IP de la interfaz de red enp0s8 con la dirección 192.168.2.10/24 modificando el archivo `/etc/network/interfaces`

```
devasc@labvm:/$ sudo nano /etc/network/interfaces
#source-directory /etc/network/interfaces.d
auto enp0s8
iface enp0s8 inet static
address 192.168.2.10
netmask 255.255.255.0
```

e. Por medio de la terminal reiniciar el servicio de red con el comando: `sudo /etc/init.d/networking restart`

f. Verificar que la IP de la interfaz enp0s8 se configuro correctamente.

```
devasc@labvm:~$ ifconfig
```

g. Iniciar el router cisco CSR1000v y configurar la IP de la interfaz Gigabit Ethernet 1 con la dirección 192.168.2.1/24.

```
Router>enable
Router#configure terminal
Router (config)#interface G1
Router (config-if)#ip address 192.168.2.1 255.255.255.0
Router (config-if)#no shutdown
Router (config-if)#exit
```

h. Verificar la conectividad entre la máquina virtual DEVASC-LABVM y el router Cisco CSR1000v.

Actividad 3: Primeros contactos con NETCONF

a. Configuración de SSH en router Cisco CSR1000v

- Crear el usuario admin con privilegio 15 (privileged EXEC).

NOTA: Recuerde que el privilegio 15 permite el acceso a todos los comandos, así como la capacidad de realizar cambios en la configuración.

```
CSR1k(config)#username admin privilege 15 password c1sco12345
```

Tome en cuenta que los usuarios creados con el comando `username` se almacenan en la base de datos de usuarios del router cisco CSR1000 (base de datos de usuarios local).

- Configurar el nombre y el dominio del router cisco CSR1000v.

```
Router(config)#hostname CSR1k
CSR1k(config)#ip domain name labo-netconf.com
```

- Generar pares de claves RSA con un tamaño de módulo de 2048 bits.

```
CSR1k(config)# crypto key generate rsa modulus 2048
```

- Habilitar SSH versión 2.

```
CSR1k(config)#ip ssh version 2
```

- Habilitar SSH en líneas VTY y configurar que el acceso a las mismas sea mediante la autenticación de usuarios locales, para ello emplear el comando `login local`.

```
CSR1k(config)#line vty 0 4
CSR1k(config-line)#transport input ssh
CSR1k(config-line)#login local
```

- Abrir una terminal en la máquina virtual DEVASC y verificar el funcionamiento de SSH.

```
devasc@labvm:~$ ssh admin@192.168.2.1
```

NOTA: La primera vez que se inicia sesión con SSH, la máquina virtual DEVASC-LABVM preguntará sobre la autenticidad del router (servidor). Debido a que se confía en el servidor, responder SI en la terminal.

- Dejar activa la sesión SSH para la siguiente parte.

b. Habilitar servicio NETCONF en router Cisco CSR1000v

NOTA: A partir de ahora y en prácticas posteriores se tratará como Administrador a la máquina virtual DEVASC-LABVM y como Agente al router Cisco CSR1000v.

- Verificar si NETCONF está activo en el Agente. Emplee la anterior sesión activa de SSH con el Agente y use el comando `show platform software yang-management process` para verificar si el demonio NETCONF SSH (ncsshd) se encuentra activo.

```
CSR1k#show platform software yang-management process
```

Debe haber obtenido una salida como la siguiente:

```
confd            : Not Running
nesd             : Not Running
syncfd           : Not Running
ncsshd           : Not Running
dmiauthd         : Not Running
nginx            : Running
ndbmand          : Not Running
pubd             : Running
```

- Emplear el comando `netconf-yang` para habilitar el servicio de NETCONF en el Agente.

```
CSR1k#configure terminal
CSR1k(config)#netconf-yang
```

- Verificar si NETCONF ya se encuentra activo en el Agente.

```
CSR1k#show platform software yang-management process
```

Debe haber obtenido una salida como la siguiente:

```
confd            : Running
nesd             : Running
syncfd           : Running
ncsshd           : Running
dmiauthd         : Running
```

```
nginx              : Running
ndbmand            : Running
pubd               : Running
```

c. Comprobar funcionamiento

- Abrir una nueva terminal en la máquina virtual DEVASC-LABVM y ejecutar NETCONF como un subsistema de SSH.

```
devasc@labvm:~$ ssh admin@192.168.2.1 -p 830 -s Netconf
```

- Debe haber obtenido una salida como la siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<capabilities>
<capability>urn:ietf:params:netconf:base:1.0</capability>
<capability>urn:ietf:params:netconf:base:1.1</capability>
... [SALIDA ELIMINADA]
<capability>urn:ietf:params:xml:ns:yang:smiv2:UDP-MIB?module=UDP-
MIB&amp;revision=2005-05-20</capability>
<capability>urn:ietf:params:xml:ns:yang:smiv2:VPN-TC-STD-MIB?module=VPN-
TC-STD-MIB&amp;revision=2005-11-15</capability>
<capability>
        urn:ietf:params:netconf:capability:notification:1.1
      </capability>
</capabilities>
<session-id>28</session-id>
</hello>]]>]]>
```

Por el momento lo único que debe tener en cuenta es que al iniciar el servicio NETCONF mediante un subsistema de SSH, el Agente automáticamente enviará un mensaje <hello> con sus capacidades y la ID para esa sesión NETCONF. Para poder establecer la sesión, el Administrador deberá enviar de la misma forma un mensaje <hello> que contendrá por lo menos la capacidad base de NETCONF (:base:1.x).

NOTA: Se suele usar la secuencia de caracteres]] >]]> para poder separar los mensajes enviados entre Administrador y Agente.

- Iniciar sesión NETCONF enviando hacia el Agente un mensaje <hello> que contenga la capacidad base de NETCONF. Para ello pegar el siguiente texto en la terminal.

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.0</capability>
  </capabilities>
</hello>]]>]]>
```

Note que el mensaje <hello> que envía el Administrador contiene solo la capacidad base de NETCONF. El URI que identifica a dicha capacidad está contenida en el elemento <capability> y corresponde al siguientes texto: *urn:ietf:params:netconf:base:1.0*

NOTA: Todos los mensajes de NETCONF como lo son <hello>, <rpc> y <rpc-reply> necesitan referenciar al XML namespace: *urn:ietf:params:xml:ns:netconf:base:1.0* para poder acceder a todos los elementos y recursos del protocolo NETCONF.

- Verificar si la sesión de NETCONF está activa. Para ello emplear el comando `show netconf-yang sessions` en la terminal que inició SSH.

```
CSR1k#show netconf-yang sessions
```

NOTA: Establecida una conexión NETCONF, el Administrador ya podrá realizar diversas solicitudes al Agente. En prácticas posteriores se tratará a profundidad este tema.

- Cerrar sesión NETCONF enviando un mensaje <rpc> con la operación <close-session>. Para ello pegar el siguiente texto en la terminal.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="121">
  <close-session/>
</rpc>]]>]]>
```

- Interprete la respuesta del Agente, no es necesario ser muy técnico.

- Verificar si la sesión NETCONF ha terminado.

- Guardar las configuraciones con el comando `wr`.

- Cerrar todas terminales de la máquina virtual DEVASC-LABVM.

2.4.1.7 Informe

a. Presentar capturas de pantalla del proceso de configuración realizado, con la debida explicación de ser necesario.

   NOTA: Algunos enunciados del procedimiento requieren respuesta o alguna evidencia de comprobación. Recuerde incluir los mismos.

b. De los dispositivos que se pueden emplear en GNS3 consulte sobre el router Cisco XRv 9000 y el switch Cisco NX-OSv 9000. Incluya los requerimientos de instalación para entornos KVM y los comandos para habilitar NETCONF. Colocar en formato de tabla.

c. Conclusiones y Recomendaciones.

2.5 Material complementario

El material adicional a las hojas guías fue desarrollado con el propósito de mejorar el aprendizaje de las temáticas abordadas en cada una de las prácticas de laboratorio. Dicho material está dirigido tanto a estudiantes como a los docentes que vean conveniente utilizar las prácticas planteadas en esté trabajo de titulación. A continuación, se describe cada uno de estos.

- **Videos**: El estudiante podrá observar la ejecución real del procedimiento de cada una de las prácticas de laboratorio. Esto ayudará a solventar cualquier duda o inconveniente que pueda llegar a presentarse al seguir el procedimiento expuesto en las hojas guías. Así también permitirá a los estudiantes obtener un mejor entendimiento de las configuraciones y actividades, logrando así facilitar su aprendizaje. El ANEXO VI contiene el link de los videos desarrollados.
- **Reactivos**: Son preguntas de opción múltiple en formato Moodle que abordan las temáticas expuestas en el marco teórico y en el trabajo preparatorio de las hojas guía. Para cada práctica existe un total de 10 reactivos, los cuales podrán ser empleados por los docentes para evaluar la correcta preparación de los estudiantes antes de la ejecución de las prácticas de laboratorio. El ANEXO VI contiene el link de los reactivos propuestos
- **Resolución de trabajos preparatorios**: Los trabajos preparatorios son actividades que necesariamente deben ser realizadas por los estudiantes como trabajo autónomo. El propósito de las mismas es introducir a los estudiantes sobre las temáticas que serán abordadas en el procedimiento de cada una de las prácticas. La resolución de los trabajos preparatorios está dirigida a los docentes. Su principal propósito es brindar una idea general de lo que se espera que los estudiantes realicen en cada trabajo preparatorio. La resolución de los trabajos preparatorios se muestra en el ANEXO VII.

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1 Resultados

Los resultados del componente desarrollado en este trabajo de titulación corresponden a la resolución de los informes de cada una de las prácticas de laboratorio. Los informes constan de tres o cuatro actividades. La primera actividad solicita mostrar las capturas de pantalla del procedimiento realizado. Los estudiantes necesariamente deberán incluir las respuestas o los análisis de resultados que son solicitados en algunos enunciados del procedimiento. La segunda y/o tercera solicita realizar una actividad teórica o práctica que permita reforzar e incrementar los conocimientos adquiridos en el desarrollo de la práctica.

La última actividad corresponde a las conclusiones y recomendaciones. El ANEXO VIII contiene la resolución detallada de cada uno de los informes. A continuación, se muestran los puntos más relevantes de la primera actividad de los informes expuestos del anexo mencionado.

3.1.1 Práctica 1: Informe

La actividad 1 denominada “Implementación del entorno de trabajo”, generó un único resultado. Al finalizar las configuraciones expuestas en esta actividad se logró de forma exitosa la instalación de la imagen de disco del router CSR1000v y la integración de la máquina virtual DEVASC con GNS3. La consideración más importante para conseguir dicho resultado fue la configuración de la GNS3 VM. Debido a que se estableció a GNS3 VM como el encargado de ejecutar las imágenes de los dispositivos (se incluye el router), fue necesario configurar los parámetros vCPU, RAM y disco duro virtual sobrepasando los requerimientos mínimos de instalación de la imagen de disco, para así evitar el mal funcionamiento del router.

En la actividad 2 denominada “Armado de topología de red y establecimiento de conexión entre dispositivos”, se obtuvo la conectividad entre la máquina virtual DEVASC y el router CSR1000v. Dicho resultado fue validado al obtener una exitosa prueba ping entre ambos dispositivos.

Para la actividad 3 con título “Primeros pasos con NETCONF”, se debe destacar tres resultados. Primero se consiguió con éxito configurar SSH en el router. Se llegó a dicha conclusión luego de establecer una conexión SSH entre la máquina virtual DEVASC y el router CSR1000v. Segundo, se habilitó exitosamente NETCONF en el dispositivo de red. Esto se aprecia claramente en la Figura 3.1 donde se observa que el demonio NETCONF SSH (ncsshd) se está ejecutando en el dispositivo, indicando que NETCONF se encuentra activo. Tercero, se logró establecer una conexión NETCONF entre cliente (máquina virtual DEVASC) y servidor (router CSR1000v). La Figura 3.2 demuestra dicho resultado. Note que en la figura la sesión NETCONF número 26 se encuentra activa, indicando que una conexión NECTONF se ha establecido correctamente. Es necesario mencionar que una sesión NETCONF se establece luego del intercambio de capacidades entre cliente y servidor.

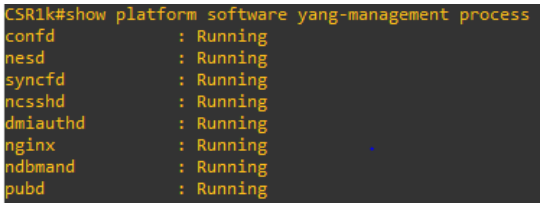


Figura 3.1. Demonio NETCONF SSH habilitado en el router CSR100v

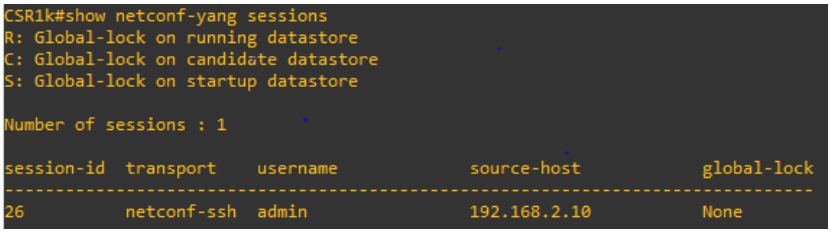


Figura 3.2. Sesión NETCONF #26 activa en el router CSR100v

3.1.2 Práctica 2: Informe

"NETCONF sobre SSH con autenticación de usuario por contraseña" es el título que corresponde a la actividad 2 del procedimiento de esta práctica. Para esta actividad se configuró el protocolo SSH para autenticar usuarios por medio de una contraseña. El método de autenticación empleado por SSH fue AAA, el cual se configuró en primer lugar para que la autenticación y la autorización de los usuarios hiciera uso de la base de datos de usuario local (contenida en el router). El resultado de dicha configuración es la Figura 3.3, que muestra una sesión NETCONF activa ejecutada por el usuario admin1, que es parte de la base de datos de usuario local. El resultado obtenido demuestra que lo único que realiza el protocolo NETCONF es confiar en la configuración de SSH mencionada anteriormente para autenticar sus usuarios NETCONF.

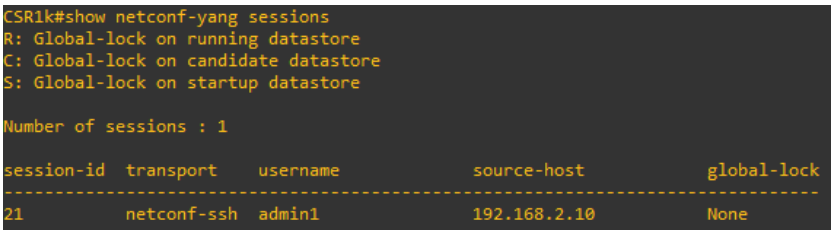


Figura 3.3 Sesión NETCONF #21 activa con usuario admin1

Existe un segundo resultado para la actividad 2, el cual se produjo al realizar un cambio en la configuración del método de autenticación de usuarios que emplea SSH. En vez de que AAA empleara una base de datos de usuario local para la autenticación y la autorización de los usuarios, se configuró para que dichos servicios hicieran uso de una base remota (contenida en un servidor TACACS+). El resultado de dicha configuración es la Figura 3.4, que muestra una sesión NETCONF activa ejecutada por el usuario gns3, que es parte de

la base de datos de usuario remota. Al igual que en el resultado anterior, se demuestra que lo único que realiza el protocolo NETCONF es confiar en la configuración SSH para autenticar sus usuarios NETCONF.

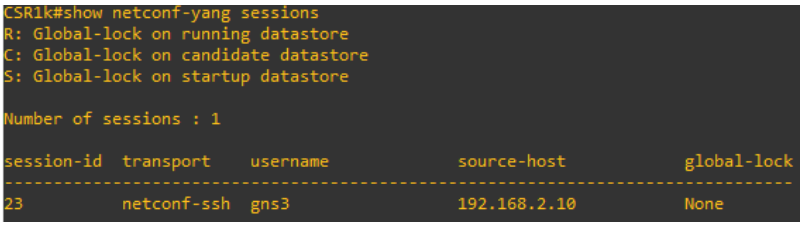


Figura 3.4 Sesión NETCONF #23 activa con usuario gns3

Para la actividad 3 con título “NETCONF sobre SSH con autenticación de usuario por clave pública”, se configuró SSH con el método de autenticación de usuario basada en RSA. El resultado de dicha configuración es la Figura 3.5, que muestra una sesión NETCONF activa ejecutada por el usuario admin2, que tiene asignada la clave pública creada en el cliente (máquina virtual DEVASC). Por última vez, se demuestra que el protocolo NETCONF se rige a la configuración de SSH para autenticar a sus usuarios NETCONF.

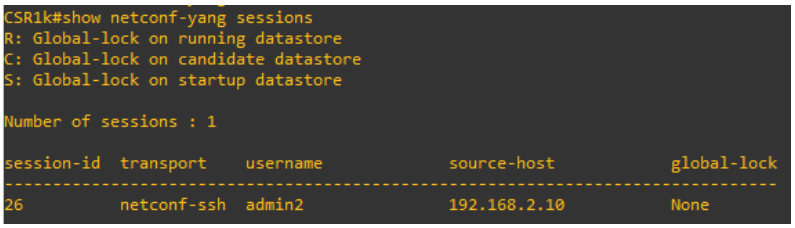


Figura 3.5 Sesión NETCONF #23 activa con usuario admin2

3.1.3 Práctica 3: Informe

En la actividad 2 denominada “Explorar modelos YANG en GITHUB”, se examinaron dos modelos de datos YANG para las interfaces de un dispositivo de red: el modelo *ietf-interfaces* y el modelo *openconfig-interfaces*. Se escogió y describió un ejemplo de nodos del lenguaje YANG (leaf, leaf-list, container y list) de cada modelo de datos. Dicha actividad permitió comprender el funcionamiento del lenguaje YANG en los modelos de datos. Por ejemplo, se entendió como un nodo container (llamado *interfaces*) se utiliza en el modelo de datos *ietf-interfaces* para brindar la estructura jerárquica de los parámetros configurables de las interfaces de un dispositivo.

Para la actividad 3 denominada “Explorar modelos YANG con pyang”, se transformó en formato árbol los modelos de datos de la actividad 2 empleando la herramienta pyang. El resultado fue una estructura de datos completa, la cual era ordenada y sencilla de comprender. Al transformar cada modelo se obtuvo una visión más amplia de toda la estructura jerárquica de los datos disponibles en los mismos. Algo que no fue posible al

examinar los modelos en el repositorio de GITHUB. Por ejemplo, se llegó a entender que *ietf-interfaces* define dos nodos container principales para separar los datos de configuración y de estado de las interfaces. Mientras que el modelo de datos *openconfig-interfaces* emplea un solo nodo container principal y es dentro del mismo donde se establecen nuevos nodos para la separación de datos de configuración y de estado de las interfaces.

"Ejemplo de la representación de datos de un dispositivo de red con YANG" es el título que corresponde a la actividad 3 del procedimiento de esta práctica. Para esta actividad se comparó el ejemplo mencionado con el modelo de datos *ietf-interfaces* y se comprobó que cada elemento XML expuesto en el ejemplo corresponde a un nodo que está definido en el modelo de datos. Esto se puede apreciar en la Figura 3.6.

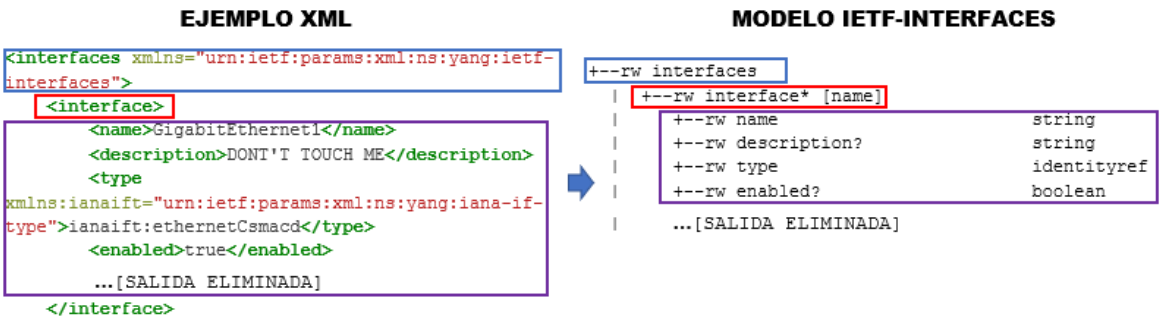


Figura 3.6. Ejemplo XML vs. Modelo ietf-interfaces

La última actividad denominada "Identificar modelos de datos YANG que permite el Agente", permitió reconocer el modelo de datos *ietf-interfaces* en el mensaje <hello> que envía el router para el intercambio de capacidades. Identificado el modelo de datos, se estableció una sesión NETCONF y posteriormente se solicitó al servidor la información de las interfaces con la estructura del modelo de datos *ietf-interfaces*. El resultado fue un texto XML con los datos de configuración de las interfaces del dispositivo que seguían la estructura jerárquica definida en el modelo de datos *ietf-interfaces*.

3.1.4 Práctica 4: Informe

La actividad 2 con título "Explorar las capacidades del Agente", permitió comprender la diferenciación entre las capacidades NETCONF y los modelos de datos YANG anunciados como capacidades que son enviados por el servidor en el mensaje <hello>. Al final de esta actividad se descubrió un total de 11 capacidades NETCONF y se concluyó la existencia de los almacenes de datos de configuración <running> y <candidate>.

La actividad 3 denominada "Enviar mensajes RPC al Agente empleando las operaciones base de NETCONF", generó una gran cantidad de resultados, los cuales se pueden

apreciar de mejor manera en el ANEXO VIII. Cada uno de los enunciados realizados en esta actividad, permitieron explicar el funcionamiento de las operaciones base del protocolo NETCONF. La Tabla 3.1 muestra a manera de resumen los puntos más destacables que se obtuvieron en los resultados al experimentar con las operaciones base.

Tabla 3.1. Resultados de la experimentación con las operaciones base

Operación	Resultados obtenidos
<get>	- Se demostró que <get> devuelve la información de configuración y de estado sólo del almacén de datos de configuración <running>.
<get-config>	- Se demostró que <get-config> devuelve solo los datos de configuración. - Cuando se habilitó el almacén de datos <candidate> en el servidor, la configuración activa de <running> fue copiada directamente en el almacén <candidate>.
<edit-config>	- <edit-config> permite realizar cambios de configuración dependiendo de su modo de operación. - <test-option> y <error-option> son parámetros opciones que permiten mejorar el rendimiento de <edit-config>.
<copy-config>	- No es posible copiar la configuración del almacén de datos <candidate> al almacén de datos <running> debido a que la capacidad Writable-running no fue anunciada por el servidor.
<delete-config>	- El único almacén de datos que no puede ser eliminado en un servidor NETCONF es el almacén <running>.
<lock>	- No importa qué tipo de cliente (NETCONF o no) intente realizar alguna configuración sobre un almacén de datos que ha sido bloqueado, de igual forma no lo conseguirá
<unlock>	- La sesión que realizó el bloqueo de los almacenes de datos, es la única que puede desbloquear los mismos.
<kill-session>	- <kill-session> permite forzar el cierre de cualquier sesión. Solo necesita el identificador de sesión NETCONF (session-id) para hacerlo.

La última actividad de esta práctica denominada “Operaciones adicionales permitidas por las capacidades anunciadas en el Agente”, genero de igual manera algunos resultados. Los mismos también se detallan de mejor manera en el ANEXO VIII. La Tabla 3.2 muestra de igual forma los puntos más relevantes de los resultados obtenidos al experimentar con algunas capacidades NETCONF anunciadas por el servidor.

Tabla 3.2. Resultados de la experimentación con algunas capacidades NETCONF

Capacidad	Resultados obtenidos

Candidate	- Se comprobó que al usar <commit> la configuración contenida en el almacén de datos de <candidate> pasa a ejecutarse como la configuración de ejecución del dispositivo. - La operación <discard-changes> no tendrá ningún efecto sobre el almacén de datos <candidate> que primero fue empleado en la operación <commit>.
Confirmed-commit:1.1	- Confirmar <commit> significa renovar la operación <commit> antes de un tiempo establecido, caso contrario <commit> se anulará. - La operación <cancel-commit> permite cancelar un <commit> que necesita confirmación.
Validate	- La operación <validate> permite verificar el contenido en un almacén de datos de configuración específico.

3.1.5 Práctica 5: Informe

Para la actividad 2 denominada “Inicio de sesión NETCONF en Agente con ncclient”, se creó un programa para iniciar una sesión NETCONF con dos métodos de autenticación: por contraseña y por clave pública. El resultado obtenido de la ejecución del programa se muestra en la Figura 3.7, que demuestra que el servidor aceptó los requerimientos para una sesión NETCONF con ambos métodos de autenticación.

```
*Apr 16 22:07:12.829: %DMI-5-AUTH_PASSED: R0/0: dmiauthd: User 'admin2' authenticated successfully from 192.168.2.10:60656 a
nd was authorized for netconf over ssh. External groups: PRIV15
*Apr 16 23:34:21.689: %DMI-5-AUTH_PASSED: R0/0: dmiauthd: User 'admin1' authenticated successfully from 192.168.2.10:60660 a
nd was authorized for netconf over ssh. External groups: PRIV15
```

Figura 3.7. Mensajes de autenticación exitosa en CLI del servidor

En la actividad 3 con título “Recuperación de información con ncclient”, se obtuvieron algunos resultados. Se debe destacar el resultado de la Figura 3.8 que muestra la salida retornada luego de ejecución de uno de los programas creados en esta actividad. Dicha figura demuestra que se logró recuperar con éxito la información de configuración de una de las interfaces del servidor.

```
devasc@labvm:~/labs/devnet-src$ /bin/python3 /home/devasc/labs/devnet-src/netconf/ncclient-get-config-int-netconf.py
Interfaz:  GigabitEthernet1
Estatus:  true
Tipo de interfaz:  ianaift:ethernetCsmacd
Dirección IPv4:  192.168.2.1
Máscara de red IPv4:  255.255.255.0
```

Figura 3.8. Salida del programa que permite recuperar la información de una interfaz

En la actividad 4 denominada “Configuración del Agente con ncclient”, se destaca el resultado de la Figura 3.9 que muestra la salida retornada luego de ejecución del programa creado para la configuración de una nueva interfaz loopback. Dicha figura demuestra que se logró con éxito realizar la configuración en el Agente.

Figura 3.9. Salida del programa que permite la creación una interfaz loopback

En la actividad 5 con título “Eliminación de información de configuración con ncclient”, se destaca el resultado de la Figura 3.11, que muestra la salida retornada luego de ejecución de uno de los programas creados en esta actividad. Dicha figura demuestra que se logró eliminar con éxito la interfaz loopback creada en la actividad 4.

Figura 3.11. Salida del programa que permite la eliminación una interfaz loopback

La última actividad de esta práctica denominada “Guardar configuración del Agente con ncclient”, se obtuvo el resultado de la Figura 3.12, que muestra que el programa creado en esta actividad ha conseguido con éxito guardar la configuración en ejecución en la configuración de inicio del Agente.

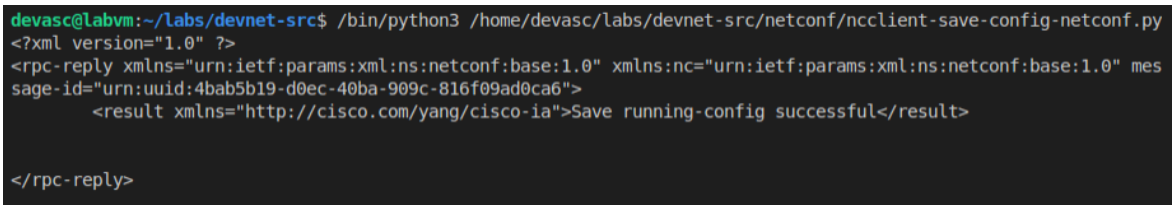


Figura 3.12. Salida del programa que permite guardar la información en la configuración de inicio

3.2 Conclusiones

- El protocolo de gestión de red NETCONF aborda las deficiencias del protocolo SNMP y la interfaz CLI para la administración/gestión de redes. NETCONF permite realizar configuraciones complejas y especializadas en los dispositivos de red, realiza distinción entre información de estado y de configuración, ofrece una conexión segura y confiable, provee una estructura de respuestas de errores, define un estándar que permite la interoperabilidad entre dispositivos de red y por último ofrece una completa y formar API permitiendo la automatización de la configuración mediante el uso de la programación. Cada una de estas características

mencionadas convierten a NETCONF en una herramienta con un gran potencial para hacer de la administración de red más simple, robusta y sobre todo efectiva.

- El lenguaje YANG estandarizado por IETF, es utilizado para modelar datos de configuración, datos de estado, RPC y notificaciones del protocolo NETCONF. Los mismos que son organizadas en módulos y submódulos para estructurar los denominados modelos de datos, que son utilizados para describir funcionalidades de un dispositivo de red. Hoy en día organizaciones como el IETF y Openconfig, así como proveedores de software y equipos de red son los encargados de crear y distribuir varios modelos de datos. Su propósito es ofrecer modelos de datos que sean capaces de cumplir las necesidades y requisitos de los administradores de red, con el objeto de facilitar la gestión de redes.

- Las prácticas de laboratorio expuestas en este trabajo de integración curricular emulan un escenario de red sencillo entre cliente y servidor, que permite demostrar la aplicación “real” de cada una de las características del protocolo de gestión de red NETCONF. Dicho conjunto de prácticas ayudará a los estudiantes de la carrera de Telecomunicaciones y afines a comprender, aprender y aplicar las funcionalidades que ofrece el protocolo NETCONF, logrando así mejorar su preparación profesional.

- Las actividades propuestas en los trabajos preparatorios, procedimiento e informes son claras y concisas, lo que permitirá una fácil comprensión por parte de los estudiantes. Esto quiere decir que los estudiantes serán capaces de resolver dichas actividades de forma sencilla y sin la existencia de inconvenientes mayores a lo normal.

- El material complementario a las hojas guía de las prácticas de laboratorio como lo son los videos, los reactivos y la resolución de informes y trabajos preparatorios, ofrece a los estudiantes y docentes una visión más amplia de los contenidos abordados en cada una de las prácticas.

- GNS3 es una herramienta muy potente, que permite emular entornos de red muy cercanos a la realidad. En este trabajo de integración curricular quedó demostrado que GNS3 cumple con las especificaciones necesarias para la creación prácticas de laboratorio que aborden el tema del protocolo NETCONF.

3.3 Recomendaciones

- Se recomienda disponer de un tamaño de memoria RAM y un almacenamiento de disco duro adecuado en el computador host, para poder ejecutar sin ningún problema los dispositivos de red empleados en las prácticas del laboratorio.

- Se recomienda explorar otros dispositivos de red o a su vez otras versiones del IOS del router Cisco CSR1000v que permitan su uso en GNS3 para la realización de las prácticas de laboratorio. De esta manera se tendrá una medida de comparación entre los distintos dispositivos de red y/o versiones, con el objeto de llegar a escoger aquel que mejor se adapte a las temáticas planteadas en las prácticas de laboratorio y sobre todo que no requiera una gran cantidad de recursos para su utilización.

- Se recomienda actualizar el contenido de las prácticas en el caso de ser necesario. Debido a que NETCONF es un protocolo relativamente nuevo y se encuentra en constante evolución, es posible que muchas de las funcionalidades expuestas en las prácticas necesiten actualizarse.

- Para futuros trabajos se recomienda explorar a detalle las notificaciones de NETCONF. De ser posible implementar una práctica referente a dicha temática.

- En el caso de disponer equipos de red (switches o routers) de forma física que soportan el protocolo NETCONF. Se recomienda adaptar las prácticas de laboratorio expuestas en este trabajo de titulación considerando dicho factor.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] A. Amarillo, “Prototipo del protocolo NETCONF para la Gestión Integrada de elementos de red,” M.S. tesis, Facultad de Ingeniería, Pontificia Universidad Javeriana, Bogotá, 2013.

[2] J. Yu and I. al Ajarmeh, “An Empirical Study of the NETCONF Protocol,” 2010 Sixth International Conference on Networking and Services, pp. 253–258, Mar. 2010.

[3] B. Wu and Y. Chang, “Integrating SNMP agents and CLI with NETCONF-based network management systems,” Proceedings - 2010 3rd IEEE International Conference on Computer Science and Information Technology, ICCSIT 2010, vol. 1, pp. 81–84, Jul. 2010.

[4] Cisco DevNet Learning Labs. (2021). Introduction to Model Driven Programmability (ex: NETCONF/YANG). [Online]. Disponible: https://developer.cisco.com/learning/modules/intro-device-level-interfaces

[5] Simple Network Management Protocol, RFC 1067, May. 1990. [Online]. Disponible: https://datatracker.ietf.org/doc/html/rfc1157

[6] J. Schonwalder, M. Bjorklund, and P. Shafer, “Network configuration management using NETCONF and YANG,” IEEE Communications Magazine, vol. 48, no. 9, pp. 166–173, Sep. 2010.

[7] Network Configuration Protocol (NETCONF), RFC 6241, Jun. 2011. [Online]. Disponible: https://datatracker.ietf.org/doc/html/rfc6241

[8] D. Valencic and V. Mateljan, “Implementation of NETCONF Protocol,” 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 421–430, May. 2019.

[9] D. Loureiro, P. Gonçlalves, and A. Nogueira, “NETCONF agent for link state monitoring,” IEEE International Conference on Communications, pp. 6565–6569, Jun. 2012.

[10] YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF), RFC 6020, Oct. 2010. [Online]. Disponible: https://datatracker.ietf.org/doc/html/rfc6020

[11] Namespaces in XML 1.0 (Third Edition), W3C Recommendation 8 December 2009, Dic. 2019. [Online]. Disponible: https://www.w3.org/TR/2009/REC-xml-names-20091208/

[12] The YANG 1.1 Data Modeling Language, RFC 7950, Agt. 2016. [Online]. Disponible: https://datatracker.ietf.org/doc/html/rfc7950

[13] J. C. Neumann, The book of GNS3: build virtual network labs using Cisco, Juniper, and more, San Francisco: No starch press, 2015, pp. 1-6.

[14] Cisco CSR 1000v and Cisco ISRv Software Configuration Guide, Cisco, San Jose, USA, Abr. 2020, pp. 109-111. Accedido: May. 28, 2022. [Online]. Disponible: https://www.cisco.com/c/en/us/td/docs/routers/csr1000/software/configuration/b_CSR1000v_Configuration_Guide.pdf

[15] GNS3. Setup wizard with the GNS3 VM. (2021). Accedido: May. 28, 2022. [Online]. Disponible: https://docs.gns3.com/docs/getting-started/setup-wizard-gns3-vm/

[16] Lab - Install The Virtual Machine Lab Environment. Cisco, 2021.

[17] “Model Driven Programmability.” developer.cisco.com. https://developer.cisco.com/site/standard-network-devices/ (accedido May 20, 2022).

[18] “Switches | Juniper Networks.” juniper.net. https://www.juniper.net/us/en/products/switches.html (accedido May 22, 2022).

[19] “Routers | Juniper Networks.” juniper.net. https://www.juniper.net/us/en/products/routers.html (accedido May 22, 2022).

[20] “Switches - Huawei Enterprise.” e.huawei.com. https://e.huawei.com/en/products/enterprise-networking/switches (accedido May 22, 2022).

[21] “Routers - Huawei Enterprise.” e.huawei.com. https://e.huawei.com/en/products/enterprise-networking/routers (accedido May 22, 2022).

[22] Using the NETCONF Protocol over Secure Shell (SSH), RFC 6242, Jun. 2011. [Online]. Disponible: https://datatracker.ietf.org/doc/html/rfc6242

[23] The Secure Shell (SSH) Transport Layer Protocol, RFC 4253, Ene. 2006. [Online]. Disponible: https://datatracker.ietf.org/doc/html/rfc4253

[24] The Secure Shell (SSH) Authentication Protocol, RFC 4252, Ene. 2006. [Online]. Disponible: https://datatracker.ietf.org/doc/html/rfc4252

[25] The Secure Shell (SSH) Connection Protocol, RFC 4254, Ene. 2006. [Online]. Disponible: https://datatracker.ietf.org/doc/html/rfc4254

[26] NETCONF Event Notifications, RFC 5277, Jul. 2008. [Online]. Disponible: https://datatracker.ietf.org/doc/html/rfc5277

[27] With-defaults Capability for NETCONF, RFC 6243, Jun. 2011. [Online]. Disponible: https://datatracker.ietf.org/doc/html/rfc6243.

[28] “Network Configuration Protocol (NETCONF) Capability URNs,” iana.org. https://www.iana.org/assignments/netconf-capability-urns/netconf-capability-urns.xhtml (accedido May 29, 2022).

[29] S. Bhushan, L. Poulopoulos and E.Nilsen-Nygaard “ncclient.” pypi.org. https://pypi.org/project/ncclient/ (accedido May 30, 2022).

[30] S.Bhushan and L.Poulopoulos. ncclient Documentation - Release 0.6.9. (2020). Accedido: May. 31, 2022. [Online]. Disponible: https://ncclient-fredgan.readthedocs.io/_/downloads/en/sphinx_version/pdf/

[31] “Introduction to ncclient.” ciscolive.com. https://yang-prog-lab.ciscolive.com/pod/0/ncclient/introduction (accessed May 30, 2022).

[32] Lab-Use NETCONF to Access an IOS XE Device. Cisco, 2021.

[33] Huawei, “Huawei NetEngine AR6700Series Enterprise Router Data Sheet,” May. 2022. [Online]. Disponible: https://e.huawei.com/es/material/networking/ar/4ff5c53bbae54c6191a248a5e5931436

[34] Cisco, “Cisco Catalyst 3650 Series Switches Data Sheet,” May, 2018. [Online]. Disponible: https://www.cisco.com/c/en/us/products/collateral/switches/catalyst-3650-series-switches/data_sheet-c78-729449.html

[35] Juniper, “ACX7100 SERIES Data Sheet,” Oct, 2021. [Online]. Disponible: https://www.juniper.net/content/dam/www/assets/datasheets/us/en/routers/acx7100-series.pdf

[36] Huawei, “What Is NETCONF?.” support.huawei.com. https://info.support.huawei.com/info-finder/encyclopedia/en/NETCONF.html (accessed May 30, 2022).

[37] CLI Book 1: Cisco ASA Series General Operations CLI Configuration Guide, 9.6, Cisco, San Jose, USA, June. 2021, pp. 907-908. Accedido: Jun. 7, 2022. [Online]. Disponible: https://www.cisco.com/c/en/us/td/docs/security/asa/asa96/configuration/general/asa-96-general-config.pdf

[38] TACACS+ Configuration Guide, Cisco, San Jose, USA, Abr. 2016, pp. 3-4. Accedido: Jun. 18, 2022. [Online]. Disponible: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/sec_usr_tacacs/configuration/xe-16/sec-usr-tacacs-xe-16-book.html

[39] “Ssh-keygen.” ssh.com. https://www.ssh.com/academy/ssh/keygen (accessed Jun. 18, 2022).

[40] H. Preston. “Native, IETF, OpenConfig... Why so many YANG models?” blogs.cisco.com. https://blogs.cisco.com/developer/which-yang-model-to-use (accedido Jun. 18, 2022).

[41] “OpenConfig-Home.” openconfig.net. https://www.openconfig.net/ (accedido Jun. 18, 2022).

[42] “pyang.” github.com. https://github.com/mbj4668/pyang (accedido Jun. 18, 2022).

[43] “27. Context Managers” book.pythontips.com. https://book.pythontips.com/en/latest/context_managers.html (accedido Jun. 18, 2022).

[44] “xml.dom.minidom — Minimal DOM implementation.” docs.python.org. https://docs.python.org/es/3/library/xml.dom.minidom.html#module-xml.dom.minidom (accedido Jun. 18, 2022).

5 ANEXOS

ANEXO I. Ejemplos de mensajes RPC empleando operaciones base.

ANEXO II. Ejemplos de capacidades NETCONF.

ANEXO III. Ejemplos de los principales nodos de datos YANG.

ANEXO IV. Lista de dispositivos que soportan NETCONF.

ANEXO V. Hojas guías de las prácticas de laboratorio.

ANEXO VI. Material digital de las prácticas de laboratorio.

ANEXO VII. Resolución de trabajos preparatorios.

ANEXO VIII. Resolución de informes.

ANEXO I

Los siguientes ejemplos fueron tomados de [10].

1. Operación <get>

Mensaje <rpc>	Mensaje <rpc-reply>

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter type="subtree">
      <top xmlns="http://example.com/schema/1.2/stats">
        <interfaces>
          <interface>
            <ifName>eth0</ifName>
          </interface>
        </interfaces>
      </top>
    </filter>
  </get>
</rpc>
```

```
<rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <top xmlns="http://example.com/schema/1.2/stats">
      <interfaces>
        <interface>
          <ifName>eth0</ifName>
          <ifInOctets>45621</ifInOctets>
          <ifOutOctets>774344</ifOutOctets>
        </interface>
      </interfaces>
    </top>
  </data>
</rpc-reply>
```

2. Operación <get-config>

Mensaje <rpc>	Mensaje <rpc-reply>

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter type="subtree">
      <top xmlns="http://example.com/schema/1.2/config">
        <users/>
      </top>
    </filter>
  </get-config>
</rpc>
```

```
<rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <top xmlns="http://example.com/schema/1.2/config">
      <users>
        <user>
          <name>root</name>
          <type>superuser</type>
          <full-name>Charlie Root</full-name>
          <company-info>
            <dept>1</dept>
            <id>1</id>
          </company-info>
        </user>
        <!-- additional <user> elements appear here... -->
      </users>
    </top>
  </data>
</rpc-reply>
```

3. Operación <edit-config>

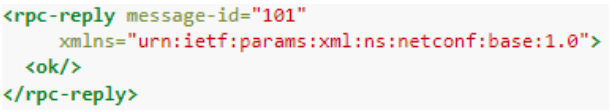
   El ejemplo mostrado a continuación emplea el atributo “operation” con el valor “delete”, para poder eliminar la configuración de "Ethernet0/0". Existen 5 valores designados a dicho atributo que permiten modificar el comportamiento de la operación <edit-config>, estos son “marge”, “replace”, “create”, “delete” y “remove”.

Mensaje <rpc>	Mensaje <rpc-reply>
	

4. Operación <copy-config>

Mensaje <rpc>	Mensaje <rpc-reply>
	

5. Operación <delete-config>

Mensaje <rpc>	Mensaje <rpc-reply>
	

6. Operación <lock>

Mensaje <rpc>	Mensaje <rpc-reply>
	

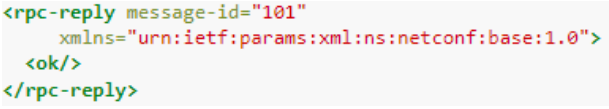
7. Operación <unlock>

Mensaje <rpc>	Mensaje <rpc-reply>
	

8. Operación <close-session>

Mensaje <rpc>	Mensaje <rpc-reply>
	

9. Operación <kill-session>

Mensaje <rpc>	Mensaje <rpc-reply>
	

ANEXO II

Los siguientes cuadros de capacidades fueron realizados tomando en cuenta [10].

Capacidad Writable-Running
Descripción
Permite que los dispositivos soporten una configuración directa sobre el almacén de datos de configuración <running>. Es decir, soportan las operaciones <edit-config> y <copy-config> cuando el parámetro target es <running>.
Identificador
urn:ietf:params:netconf:capability:writable-running:1.0
Nuevas Operaciones
Ninguna

Capacidad Candidate	
Descripción	
Permite que los dispositivos empleen el almacén de datos de configuración <candidate>.	
Identificador	
urn:ietf:params:netconf:capability:candidate:1.0	
Nuevas Operaciones	

<commit>
Permite establecer la configuración contenida en el almacén de datos de configuración <candiate> como la configuración en ejecución del dispositivo.
	Ejemplo
	<rpc>
	`<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <commit/> </rpc>`
	<rpc-reply>
	`<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <ok/> </rpc-reply>`

<discard-changes>
En el caso de que el cliente ya no requiera emplear la operación <commit>, puede emplear <discard-changes> para resetear el almacén de datos de configuración <candidate> y que el contenido del mismo sea igual a del almacén de datos <running>.
	Ejemplo
	<rpc>
	`<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <discard-changes/> </rpc>`
	<rpc-reply>
	`<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <ok/> </rpc-reply>`

Capacidad Confirmed Commit

Descripción

Permite confirmar la operación <commit>, lo que implica que <commit> se mantendrá activa en el dispositivo solo si se confirma dicha operación, caso contrario se desactiva.

Identificador

urn:ietf:params:netconf:capability:confirmed-commit:1.1

Dependencia

La capacidad es relevante si el dispositivo soporta la capacidad Candidate

Nuevas Operaciones

<cancel-commit>

Permite cancelar una confirmación de la operación <commit> en curso.

Parámetros

persist-id: El valor deber ser igual al parámetro <persit> de la operación <commit> para poder cancelar la confirmación.

Ejemplo

<rpc>

```
<rpc message-id="102"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <cancel-commit/>
</rpc>
```

<rpc-reply>

```
<rpc-reply message-id="102"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

Modificación de Operaciones existentes

Nuevos parámetros de <commit>

confirmed: Habilita la confirmación de <commit>.

confirm-timeout: Periodo de tiempo límite en segundos para poder confirmar <commit>.

persist: Establece un token para la operación <commit> confirmada.

persist-id: Corresponde al valor configurado en <persist>. Se emplea para realizar seguimiento a la operación <commit> confirmada.

Ejemplo

```
<!-- start a persistent confirmed-commit -->
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <commit>
    <confirmed/>
    <persist>IQ,d4668</persist>
  </commit>
</rpc>

<rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>

<!-- confirm the persistent confirmed-commit,
     possibly from another session -->
<rpc message-id="102"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <commit>
    <persist-id>IQ,d4668</persist-id>
  </commit>
</rpc>

<rpc-reply message-id="102"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

Capacidad Rollback-on-Error

Descripción

Permite el valor “rollback-on-error”.

Identificador

urn:ietf:params:netconf:capability:rollback-on-error:1.0

Modificación de Operaciones existentes

<edit-config>

Admite el valor “rollback-on-error” para el parámetro <error-option> de la operación <edit-config>.

Este valor permite que en el caso de un error ocurra, el servidor detendrá el procesamiento de la operación <edit-config> y restaurará la configuración especificada a su estado inicial.

Ejemplo

<rpc>

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <error-option>rollback-on-error</error-option>
    <config>
      <top xmlns="http://example.com/schema/1.2/config">
        <interface>
          <name>Ethernet0/0</name>
          <mtu>100000</mtu>
        </interface>
      </top>
    </config>
  </edit-config>
</rpc>
```

<rpc-reply>

```
<rpc-reply message-id="102"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

Capacidad Validate

Descripción

Permite verificar si en una configuración existen errores sistemáticos y semánticos antes de aplicar la configuración en el dispositivo.

Identificador

urn:ietf:params:netconf:capability:validate:1.1

Nuevas Operaciones

<validate>

Valida el contenido de una configuración específica.

Parámetros

source: Nombre del almacén de datos de configuración a validar.

Ejemplo

<rpc>

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <validate>
    <source>
      <candidate/>
    </source>
  </validate>
</rpc>
```

<rpc-reply>

```
<rpc-reply message-id="102"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

Modificación de Operaciones existentes

Permite que la operación <edit-config> acepte el parámetro <test-option>

Capacidad Startup
Descripción
Permite que los dispositivos empleen el almacén de datos de configuración <startup>.
Identificador
urn:ietf:params:netconf:capability:startup:1.0
Nuevas Operaciones
Ninguna

ANEXO III

Los siguientes ejemplos fueron tomados de [10].

NODO LEAF

YANG	XML NETCONF

`leaf host-name {`
`  type string;`
`  description "Hostname for this system";`
`}`
	`<host-name>my.example.com</host-name>`

NODO LEAF-LIST

YANG	XML NETCONF

`leaf-list domain-search {`
`  type string;`
`  description "List of domain names to search";`
`}`
	`<domain-search>high.example.com</domain-search>`
	`<domain-search>low.example.com</domain-search>`
	`<domain-search>everywhere.example.com</domain-search>`

NODO CONTAINER

YANG	XML NETCONF

`container system {`
`  container login {`
`  leaf message {`
`  type string;`
`  description`
`  "Message given at start of login session";`
`  }`
`  }`
`}`
	`<system>`
	`  <login>`
	`  <message>Good morning</message>`
	`  </login>`
	`</system>`

NODO LIST

YANG	XML NETCONF

`list user {`
`  key "name";`
`  leaf name {`
`  type string;`
`  }`
`  leaf full-name {`
`  type string;`
`  }`
`  leaf class {`
`  type string;`
`  }`
`}`
	`<user>`
	`  <name>glocks</name>`
	`  <full-name>Goldie Locks</full-name>`
	`  <class>intruder</class>`
	`</user>`
	`<user>`
	`  <name>snowey</name>`
	`  <full-name>Snow White</full-name>`
	`  <class>free-loader</class>`
	`</user>`
	`<user>`
	`  <name>rzell</name>`
	`  <full-name>Rapun Zell</full-name>`
	`  <class>tower</class>`
	`</user>`

ANEXO IV

Según [17], los dispositivos CISCO que soportan NETCONF son los siguientes.

CISCO			
Routers	IOS XE		IOS-XR
	Serie ASR 1000 Serie ISR 4000 CSR 1000v ISRv Serie ASR 900 ASR 920	Serie ISR 1000 IR 1101 Series	CRS XRv 9000 NCS 5500 NCS 5000 ASR 9000(32-64 bit) NCS 6000
Switches	IOS XE	NX-OS	
	Catalyst 4500 Catalyst 3650 Catalyst 3850 Catalyst 9000 Serie NCS 4200 Serie IE 3x00	Nexus 3000 Nexus 9000 Nexus 6000	

Según [18], [19], los dispositivos JUNIPER que soportan NETCONF son los siguientes.

JUNIPER			
Routers	Serie MX2000 MX960 MX480 MX240 MX204 MX150 MX104	Serie ACX7100 Serie ACX6000 Serie ACX5000 Serie ACX500 Serie ACX1000 Serie ACX4000 Serie ACX2000	ACX710 PTX1000 PTX1001-36MR PTX1002 PTX1003
Switches	Serie EX9251 Serie EX9200 QFX5700 QFX5130 QFX5220		

Según [20], [21], los dispositivos HUAWEI que soportan NETCONF son los siguientes.

HUAWEI	
Routers	Serie NetEngine AR6700 Serie NetEngine AR6000 Serie NetEngine AR8000 Serie NetEngine AR650

	Serie NetEngine AR610	
Switches	Serie CloudEngine S12700E Serie CloudEngine S8700 Serie CloudEngine S6730-H Serie CloudEngine S5731-H Serie CloudEngine S5736-S Serie CloudEngine S5735-S	Serie CloudEngine S5735-S-IA Serie CloudEngine S5735-L-I Serie CloudEngine S5735-S-I Serie S5720I-SI

ANEXO V

PRÁCTICA N° 2

1. TEMA

CAPA DE TRANSPORTE DE NETCONF

2. OBJETIVOS

2.1. Ejecutar el protocolo de gestión de red NETCONF como un subsistema SSH.
2.2. Configurar dos formas de autenticación de usuarios NETCONF: por contraseña y por clave pública.

3. MARCO TEÓRICO

3.1. SHH (Secure Shell)

SSH es un protocolo seguro para acceso remoto basado en una arquitectura de cliente-servidor. Una sesión SSH se establece cuando se cumplen los siguientes pasos. Primero el cliente ejecutará una conexión de transporte SSH empleando el protocolo de la capa de transporte SSH. Segundo, el cliente invocará el protocolo de autenticación SSH para autenticar al usuario. Finalmente, si la autenticación fue exitosa el cliente invocará al protocolo de conexión SSH que dará como resultado una sesión SSH [22]. Cada uno de los protocolos mencionados se describen a continuación:

- Protocolo de la capa de transporte SSH: Proporciona cifrado fuerte, autenticación de host criptográfico y protección de integridad. Este protocolo es el encargado de permitir la negociación de los siguientes parámetros: método de intercambio de claves, algoritmo de clave pública, algoritmo de cifrado simétrico, algoritmo de autenticación de mensajes y algoritmo hash [23].

- Protocolo de autenticación SSH: Define los métodos de autenticación de cliente como por ejemplo clave pública o contraseña [24].

- Protocolo de conexión SSH: Proporciona sesiones de inicio de sesión interactivas, ejecución remota de comandos, conexiones TCP/IP reenviadas y conexiones X11 reenviadas [25].

3.2. NETCONF sobre SSH

NETCONF se ejecuta sobre SSH como un subsistema SSH denominado “netconf”. Es importante mencionar que la compatibilidad con subsistemas es una característica netamente de SSH versión 2 (SSHv2), lo que implica que cualquier implementación de NETCONF sobre SSHv1 no sería posible [22]**.**

Debido a que SSH es el responsable de autenticar sesiones SSH, el nombre de usuario proporcionado por la sesión SSH será empleado por la capa de mensajes de NETCONF, como el nombre de usuario de NETCONF sin ninguna modificación [22]**.**

Un servidor NETCONF debe proporcionar acceso de forma predeterminada al subsistema SSH “netconf” cuando la sesión SSH se establezca utilizando el puerto TCP 830 asignado por IANA. Aunque los servidores pueden ser configurados para permitir el acceso al subsistema NETCONF SSH a través de otros puertos [22]**.**

Un usuario o aplicación podría emplear la siguiente línea de comando para invocar NETCONF como un subsistema SHH:

```
[usuario@cliente]$ ssh -s server.example.org -p 830 netconf
```

La opción -s hace que el comando "netconf" se invoque como un subsistema SSH y la opción -p indica el puerto predeterminado de este subsistema [22]**.**

4. TRABAJO PREPARATORIO

4.1. Revisar el marco teórico para la realización de la práctica.

4.2. Consultar y responder a las siguientes preguntas.

- ¿NETCONF podría superponerse sobre otro protocolo de transporte a parte de SSH?
- ¿Quién es responsable de la autenticación de usuarios NETCONF?
- ¿Qué es AAA (Authentication, Authorization, Accounting)?
- ¿Qué es un servidor TACACS+?
- ¿Qué es ssh-keygen? y ¿Cómo se emplea para la autenticación de usuario con SSH?

5. EQUIPO Y MATERIALES

- PC (GNS3)

6. PROCEDIMIENTO

6.1. Topología de red

a. Implementar la topología de red mostrada en la Figura 1.
NOTA: Puede emplear el mismo proyecto de GNS3 creado en la práctica 1.

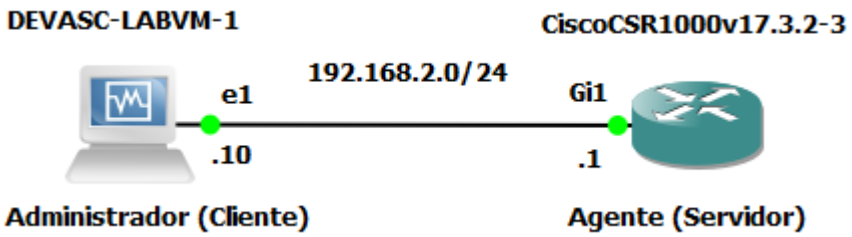


Figura 1. Topología de red

IMPORTANTE: NETCONF depende de cómo esté configurado SSH para autenticar y ofrecer seguridad a las sesiones NETCONF. Los siguientes literales muestran dos formas de autenticación que emplea SSH para autenticar sesiones NETCONF.

6.2. NETCONF sobre SSH con autenticación de usuario por contraseña

NOTA: La secciones A y B se podrían omitir ya que fueron realizadas en la práctica 1. Se recomienda verificar las mismas.

A. Configurar autenticación del dispositivo

a. Configurar el nombre y el dominio del dispositivo.

NOTA: Tanto el nombre del dispositivo como el nombre del dominio son empleados como parte de la clave criptográfica cuando esta se genera. Es por esta razón que ambos nombres deben registrarse antes de emitir el comando `crypto key`.

```
Router(config)#hostname CSR1k
CSR1k(config)#ip domain name labo-netconf.com
```

b. Generar pares de claves RSA con un tamaño de módulo de 2048 bits.

```
CSR1k(config)# crypto key generate rsa modulus 2048
```

B. Habilitar SSH versión 2

```
CSR1k(config)#ip ssh version 2
```

C. Configurar autenticación con base de datos de usuarios local

a. Configurar AAA en el Agente.

Una de las características de NETCONF es que sigue de cerca la funcionalidad nativa de los dispositivos, permitiendo de esta forma que el protocolo de transporte (en este caso SSH) haga uso de los métodos de autenticación disponible en el Agente sin ningún inconveniente. En los siguientes literales configurará AAA que es una característica nativa del Agente.

i. Crear usuario con el mayor privilegio (15). Recuerde que los usuarios creados se almacenan en la base de datos local del dispositivo.

NOTA: Los usuarios NETCONF deben poseer el privilegio 15 para poder iniciar el servicio NETCONF, caso contrario no se establecerá la conexión.

```
CSR1k>enable
CSR1k#configure terminal
CSR1k(config)#username admin1 privilege 15 password
c1sco12345
```

ii. Habilitar AAA.

```
CSR1k(config)#aaa new-model
```

iii. Definir la lista de autenticación de inicio de sesión predeterminada con el método local.

```
CSR1k(config)#aaa authentication login default local
```

iv. Definir la lista de autorización EXEC predeterminada con el método local.
NOTA: El usuario NETCONF debe ser autorizado por el servidor para poder ejecutar una shell EXEC, caso contrario el usuario no podrá acceder al servicio NETCONF.

```
CSR1k(config)#aaa authorization exec default local
```

D. Configurar líneas VTY

a. Acceder a las líneas VTY.

```
CSR1k(config)#line vty 0 4
```

b. Habilitar SSH como protocolo de acceso a las líneas VTY

```
CSR1k(config-line)#transport input ssh
```

c. Aplicar la lista de métodos autenticación predeterminada en las líneas VTY.

```
CSR1k(config-line)#login authentication default
```

E. Iniciar sesión NETCONF empleando la base datos de usuarios local

a. Acceder al proceso de NETCONF a través de SSH.

Un usuario o aplicación deben invocar a NETCONF como subsistema SSH en el puerto TCP 830 asignado por la IANA. La siguiente línea de comando es la forma más común para iniciar el servicio NETCONF.

```
devasc@labvm:~$ ssh admin1@192.168.2.1 -p 830 -s netconf
```

b. Enviar mensaje <hello> al Agente. Para ello pegar el siguiente texto en la terminal.

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.0</capability>
  </capabilities>
</hello>]]>]]>
```

c. Verificar la sesión NETCONF y luego cerrar sesión. Recuerde emplear el siguiente texto para cerrar la sesión.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-
id="121">
<close-session/>
</rpc>]]>]]>
```

F. Configurar autenticación con base de datos de usuarios remota (servidor TACACS+)

AAA suele usarse comúnmente con bases de datos de usuarios remotos. Lo que implica que las sesiones NETCONF pueden ser autenticadas empleando una base remota.

a. Descargar e instalar el AAA Docker appliance en GNS3. (ver Figura 2)

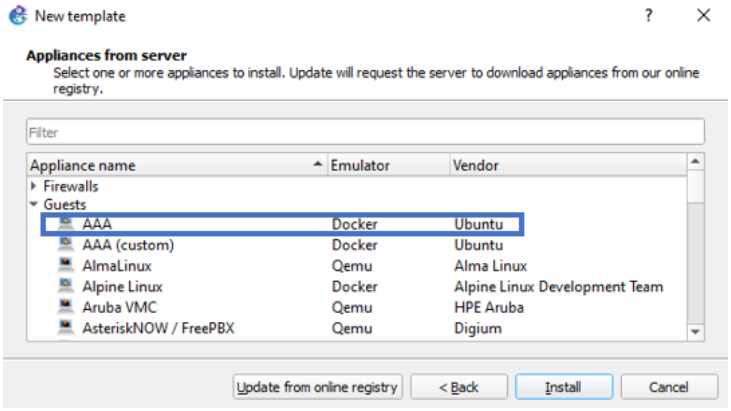


Figura 2. Ventana para escoger el AAA Docker appliance

b. Arrastrar el AAA Docker appliance a la ventana central de GNS3 y conectar su interfaz eth0 a la interfaz Gigabit Ethernet 2 del router. (ver Figura 3)

Figura 3. Topología de red con AAA Docker appliance

c. Configurar la interface Gigabit Ethernet 2 del router.

```
CSR1k(config)#interface G2
CSR1k(config-if)#ip address 192.168.3.1 255.255.255.0
CSR1k(config-if)#no shutdown
CSR1k(config-if)#exit
```

d. Configurar la interfaz etho del AAA Docker appliance. Para ello modificar el archivo `/etc/network/interfaces`.

```
root@AAA-1:~# nano /etc/network/interfaces
#
# This is a sample network config uncomment lines to configure
the network
#

# Static config for eth0
auto eth0
iface eth0 inet static
        address 192.168.3.100
```

```
        netmask 255.255.255.0
#       gateway 192.168.0.1
#       up echo nameserver 192.168.0.1 > /etc/resolv.conf

# DHCP config for eth0
# auto eth0
# iface eth0 inet dhcp
```

e. Reiniciar AAA Docker appliance y verificar la configuración IP.
f. Verificar la base de datos almacenada en AAA Docker appliance.

NOTA: Para esta parte se hará uso del servidor TACACS+ contenido en el AAA Docker appliance. El archivo de configuración de este servicio es el siguiente: `/etc/tacacs+/tac_plus.conf`

```
root@AAA-1:~# nano /etc/tacacs+/tac_plus.conf
```

Note que el archivo contiene algunas configuraciones. Muchas de ellas son necesarias entender para poder configurar AAA en el router. Estas se detallan a continuación:

- El servidor TACACS+ posee una clave, la cual debe ser usada por los clientes para acceder al mismo. Note que en este caso la clave es “gns3”, por tanto, debe ser empleada por el router para poder acceder al servidor TACACS+. (ver Figura 4)

```
accounting file = /var/log/tac_plus.acct

# This is the key that clients have to use to access Tacacs+

key = gns3
```

Figura 4. Clave de clientes para el acceso a TACACS+

- En el servidor TACACS+ se encuentra configurado el usuario gns3, al cual se le atribuye un nombre, un grupo y una clave para el inicio de sesión. (ver Figura 5)

```
user = gns3 {
    name = "Admin User"
    member = admin
    login = des AxKP5aUynXxrg
                service = junos-exec {
                        local-user-name = remote-admin
        }
}
```

Figura 5. Usuario gns3 en servidor TACACS+

- El usuario gns3 pertenece al grupo admin que es el grupo con el mayor privilegio (15). (ver Figura 6)

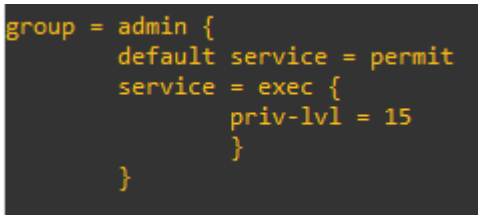


Figura 6. Grupo admin en servidor TACACS+

g. Eliminar la lista de métodos predeterminada para la autenticación de inicio de sesión y autorización EXEC.

```
CSR1k(config)#no aaa authentication login default local
CSR1k(config)#no aaa authorization exec default local
```

h. Configurar AAA con servidor remoto TACACS+.

i. Habilitar AAA

```
CSR1k(config)#aaa new-model
```

ii. Configurar el grupo de servidores TACACS+ con el nombre `romotelist`.

```
CSR1k(config)#aaa group server tacacs+ remotelist
```

iii. Crear un servidor con el nombre `container`.

```
CSR1k(config-sg-tacacs+)#server name container
CSR1k(config-sg-tacacs+)#exit
```

iv. Establecer como un servidor TACACS+ al servidor creado en el paso anterior.

```
CSR1k(config)#tacacs server container
```

v. Configurar dirección IP del servidor TACACS+.

```
CSR1k(config-server-tacacs)#address ipv4 192.168.3.100
```

vi. Configurar la clave de acceso (key) del servidor TACACS+.

```
CSR1k(config-server-tacacs)#key gns3
```

i. Definir la lista de métodos predeterminada para la autenticación de inicio de sesión y autorización EXEC con el grupo de servidores TACACS+ como método principal y el local como el método secundario.

```
CSR1k(config)#aaa authentication login default group remotelist
local
```

```
CSR1k(config)#aaa authorization exec default group remotelist
local
```

NOTA: El grupo `remotelist` contiene al servidor TACACS+ de la topología. Por tanto, la lista predeterminada hará uso primero de la base de datos de usuario remota almacenada en el servidor TACASC+

G. Iniciar sesión NETCONF empleando la base datos de usuarios remota

a. Activar debug, para verificar el funcionamiento de la autenticación y autorización AAA con el servidor remoto TACACS+.

```
CSR1k#debug aaa authentication
CSR1k#debug aaa authorization
```

b. Acceder al proceso de NETCONF a través de SSH con usuario gns3. Comente los resultados obtenidos por el debug.

NOTA: El usuario gns3 tiene asociada la contraseña gns3.

```
devasc@labvm:~$ ssh gns3@192.168.2.1 -p 830 -s netconf
```

c. Enviar mensaje <hello> al Agente. Para ello pegar el siguiente texto XML en la terminal.

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.0</capability>
  </capabilities>
</hello>]]>]]>
```

d. Verificar la sesión NETCONF y luego cerrar sesión. Recuerde emplear el siguiente texto XML para cerrar la sesión.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-
id="121">
<close-session/>
</rpc>]]>]]>
```

e. Apagar en AAA Docker appliance y acceder al proceso de NETCONF con usuario admin1. Comente los resultados obtenidos por el debug.

```
devasc@labvm:~$ ssh admin1@192.168.2.1 -p 830 -s netconf
```

f. Repita los pasos c y d.
g. Desactivar debug.

```
CSR1k#no debug aaa authentication
CSR1k#no debug aaa authorization
```

6.3. NETCONF sobre SSH con autenticación de usuario por clave pública

En esta parte ya no se hará uso del AAA Docker appliance, razón por la cual debe eliminarlo de la topología de red. Hacer uso de la topología mostrada en la Figura 1.

A. Configurar autenticación del dispositivo

NOTA: no es necesario volver configurar la autenticación del dispositivo, ya que fue realizado en el punto 6.2.

B. Configurar autenticación de usuario con clave pública

a. Generar un par de claves (pública y privada) en la máquina virtual DEVASC-LABVM (Administrador).

```
devasc@labvm:~$ssh-keygen -t rsa -b 2048
Enter file in which to save the key (/home/devasc/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):c1sco123
Enter same passphrase again:c1sco123
```

NOTA: La contraseña que se empleó para la creación del par de claves fue c1sco123.

b. Verificar la creación de claves. Para ello revisar la existencia de los archivos `id_rsa` y `id_rsa.pub` en el directorio `~/.shh`

```
devasc@labvm:~$ cd .ssh
devasc@labvm:~/.ssh$ ls
config  id_rsa  id_rsa.pub  known_hosts
```

c. Visualizar el contenido de claves. Tome en cuenta que el archivo `id_rsa` contiene la clave privada y el `id_rsa.pub` la clave pública.

```
devasc@labvm:~/.ssh$ cat id_rsa
devasc@labvm:~/.ssh$ cat id_rsa.pub
```

d. Copiar clave pública del Administrador.

NOTA: Es necesario separar la clave pública en 70 caracteres por línea, esto para evitar errores al pegar la clave en el Agente. Puede emplear el siguiente comando: `fold -b -w 70 ~/.ssh/id_rsa.pub`

Un ejemplo de la clave pública es el siguiente:

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC+997simmeFk3JFx03xdHZ1e5fhNWpvj
47puBxWhAAI7g0wFAASPk8FFBBBOALanT3VqQE5eSkT3GYlF/3LiEwg/H25huc30Y4iMHL
```

```
eSfwi53vJoMzFXSgudP3mJSX+HqD50Cw7fzBstEmQIUlgRqA6DFkEQE9yXWZpWEiLNmFet
NrtFagC70+iB4YX9r7Fa9H6pqgG4r1dchX29nRW2ja+xKiT8y+JBdeq6CT+IEIKgFJm+bW
+E4Peuf03hO6ahxFsaSpO5Ur6cKRlNPidPd4nTIzxTmgh5WR9aagBcNZ2Aj88NL++GY4Ye
OTvlrVtXtqKbNkmzqILgyuIAzvyeof devasc@labvm
```

e. Configurar SSH en el Agente para autenticación de usuario basado en RSA.

```
CSR1k(config)# ip ssh pubkey-chain
CSR1k(conf-ssh-pubkey)# username admin2
CSR1k (conf-ssh-pubkey-user)# key-string
CSR1k (conf-ssh-pubkey-data)# PEGAR CLAVE PÚBLICA
(ADMINISTRADOR)
CSR1k (conf-ssh-pubkey-data)#exit
```

NOTA: La máquina virtual puede tener tantos pares de claves como usuarios que desee que accedan al servicio NETCONF. Note que clave pública de la máquina virtual está asociada solo al usuario admin2.

f. Verificar que la clave púbica del Administrador sea la correcta. Emplee el comando: `ssh-keygen -l -f id_rsa.pub -E md5` y compare la salida con clave almacenada en la configuración del Agente.

NOTA: La autenticación del usuario será exitosa si la clave pública almacenada en el Agente es verifica con la clave pública almacenada en el Administrador.

g. Crear el usuario admin2 con privilegio 15 en la base de datos local del Agente.

```
CSR1k(config)#username admin2 privilege 15
```

C. Configurar líneas VTY

NOTA: No es necesario volver a realizar las configuraciones en las líneas VTY. Es importe mencionar que el comando `login authentication default` no tendrá ningún efecto cuando se emplee el usuario admin2 para el acceso SSH.

D. Iniciar sesión NETCONF

a. Acceder al proceso de NETCONF a través de SSH con usuario admin2.

```
devasc@labvm:~$ ssh admin2@192.168.2.1 -p 830 -s netconf
```

NOTA: Al acceder al servicio NETCONF con admin2 por primera vez, se requiera escribir la contraseña con la que se creó el par de claves. Luego de esto no se volverá a solicitar dicha clave, a menos que reinicie o apague la máquina virtual.

b. Iniciar sesión NETCONF. Para ello pegar el siguiente texto XML en la terminal.

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
```

```
    <capability>urn:ietf:params:netconf:base:1.0</capability>
  </capabilities>
</hello>]]>]]>
```

c. Verificar la sesión NETCONF y luego cerrar sesión. Recuerde emplear el siguiente texto XML para cerrar la sesión.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-
id="121">
  <close-session/>
</rpc>]]>]]
```

d. Guardar la configuración realizada en el Agente, ya que será empleada en prácticas posteriores.

NOTA: Emplear el comando wr en la CLI del Agente.

7. INFORME

7.1. Presentar capturas de pantalla del proceso de configuración realizado, con la debida explicación de ser necesario.
NOTA: Algunos enunciados del procedimiento requieren alguna evidencia de comprobación. Recuerde incluir los mismos.

7.2. Realice un cuadro comparativo entre la autenticación de usuario por contraseña y por clave pública.

7.3. Describa de forma concisa las consideraciones de seguridad establecidas en el RFC 6242.

7.4. Conclusiones y Recomendaciones.

8. REFERENCIAS

Las referencias de la práctica 2 se encuentran en el apartado de referencias de este documento.

PRÁCTICA N° 3

1. TEMA

LENGUAJE DE MODELADO DE DATOS YANG

2. OBJETIVOS

2.1. Comprender que es lenguaje YANG y que son los modelos de datos YANG.
2.2. Utilizar pyang para explorar modelos de datos YANG.
2.3. Comprender como los modelos de datos YANG son usados en un dispositivo de red cuando se envía o recibe datos.

3. MARCO TEÓRICO

3.1. Lenguaje YANG

YANG es un lenguaje de modelado de datos diseñado específicamente para modelar datos de configuración y datos de estado, RPC (Remote Procedure Calls) y notificaciones del protocolo de gestión de red NETCONF. Aunque, YANG también puede ser usado por otros protocolos como RESTCONF y CoAP [12].

YANG es un lenguaje estructurado y fuertemente tipado, características que son beneficiosas para un lenguaje de modelado de datos [4]. Algunos de los aspectos más importantes del lenguaje YANG, se mencionan a continuación:

- YANG organiza los datos en forma jerárquica como un árbol en el que cada nodo tiene un nombre y un valor o un conjunto de nodos secundarios.
- Un módulo es un único modelo de datos, que define una jerarquía de datos.
- YANG estructura los modelos de datos en módulos y submódulos.
- Una estructura de datos puede estar definida dentro un módulo o puede ser importada de otros.
- Se usan nodos `leaf` para representar un único atributo/dato. (ver Figura 1)
- Cada nodo `leaf` debe tener asociado un tipo integrado YANG (ver Figura 2) que es establece al emplear la declaración `type`.
- Se usan nodos `leaf-list` para representar una secuencia de nodos `leaf` del mismo tipo. (ver Figura 3)
- Se usan nodos `contaniner` para agrupar nodos que tienen relación entre sí. (ver Figura 4)
- Se usan nodos `list` para crear una secuencia de n listas.
- Una lista es como un nodo container y se identifica de forma única por valor de `key`. (ver Figura 5)

```
leaf host-name {
    type string;
    description "Hostname for this system";
}
```

Figura 1. Ejemplo de nodo leaf

Name	Description
binary	Any binary data
bits	A set of bits or flags
boolean	"true" or "false"
decimal64	64-bit signed decimal number
empty	A leaf that does not have any value
enumeration	One of an enumerated set of strings
identityref	A reference to an abstract identity
instance-identifier	A reference to a data tree node
int8	8-bit signed integer
int16	16-bit signed integer
int32	32-bit signed integer
int64	64-bit signed integer
leafref	A reference to a leaf instance
string	A character string
uint8	8-bit unsigned integer
uint16	16-bit unsigned integer
uint32	32-bit unsigned integer
uint64	64-bit unsigned integer
union	Choice of member types

Figura 2. Tipos integrados de YANG

```
leaf-list domain-search {
    type string;
    description "List of domain names to search";
}
```

Figura 3. Ejemplo de nodo `leaf-list`

```
container system {
    container login {
        leaf message {
            type string;
            description
                "Message given at start of login session";
        }
    }
}
```

Figura 4. Ejemplo de nodo `container`

```
list user {
    key "name";
    leaf name {
        type string;
    }
    leaf full-name {
        type string;
    }
    leaf class {
        type string;
    }
}
```

Figura 5. Ejemplo de nodo `list`

3.2. ¿Qué es un modelo de datos?

Un modelo de datos es un método acordado y bien definido que permite describir “algo” [4]. Por ejemplo, considere el siguiente modelo de datos simple para un automóvil.

- Automóvil
    - Marca: Ford, Audi, u otro.
    - Color: Azul, negro u otro.
    - Transmisión automática: SI o NO
    - Precio: 24.000 $ u otro.
    - Año de fabricación: 2019, 2021 u otro.

Emplear dicho modelo de datos genérico, permitirá describir un automóvil de tal forma que sea entendible para todos.

3.3. Modelo de datos YANG

Un modelo de datos YANG describe cualquier aspecto relacionado a redes. Puede describir una función de un dispositivo de red o describir algún servicio de red. Dichos modelos están destinados a ser empleados por sistemas de administración u orquestación de redes [4]. En NETCONF los modelos de datos son utilizados por administradores de redes o aplicaciones para manipular los datos de los dispositivos.

3.4. XML

XML sirve como formato de codificación para el protocolo NETCONF. Su función es establecer una jerarquización compleja de datos, que sea capaz de expresarse en un formato de texto que pueda ser leído, guardado y manipulado ya sea con herramientas de texto tradicionales o con herramientas específicas de XML [12].

La sintaxis, así como la codificación XML es empleada en las RPC de la capa de mensajes, en las operaciones del protocolo y en los módulos YANG de la capa de contenido [7] [10]. Esto quiere decir que los módulos YANG se pueden traducir a una sintaxis XML. A continuación, se muestra la correspondiente codificación XML de los ejemplos mostrados en 3.2.

```
<host-name>my.example.com</host-name>
```

Figura 6. XML NETCONF de nodo `leaf`

```
<domain-search>high.example.com</domain-search>
<domain-search>low.example.com</domain-search>
<domain-search>everywhere.example.com</domain-search>
```

Figura 7. XML NETCONF de nodo `leaf-list`

```
<system>
  <login>
    <message>Good morning</message>
  </login>
</system>
```

Figura 8. XML NETCONF de nodo `container`

```
<user>
  <name>glocks</name>
  <full-name>Goldie Locks</full-name>
  <class>intruder</class>
</user>
<user>
  <name>snowey</name>
  <full-name>Snow White</full-name>
  <class>free-loader</class>
</user>
<user>
  <name>rzell</name>
  <full-name>Rapun Zell</full-name>
  <class>tower</class>
</user>
```

Figura 9. XML NETCONF de nodo `list`

4. TRABAJO PREPARATORIO

4.1. Revisar el marco teórico para la realización de la práctica.

4.2. Responda a los siguientes enunciados:

- Describa los modelos de datos YANG de IETF, OPENCONFIG y nativos.
- ¿Qué es un espacio de nombres XML (XML namespace) y qué relación tiene con un módulo YANG?
- ¿Qué es y para que se emplea declaración YANG: `prefix`?
- ¿Qué es pyang y para qué es empleado?

5. EQUIPO Y MATERIALES

- PC (GNS3)

6. PROCEDIMIENTO

6.1. Topología de red

a. Implementar la topología de red mostrada en la Figura 10.
NOTA: Emplear el proyecto de GNS3 creado en la práctica 2.

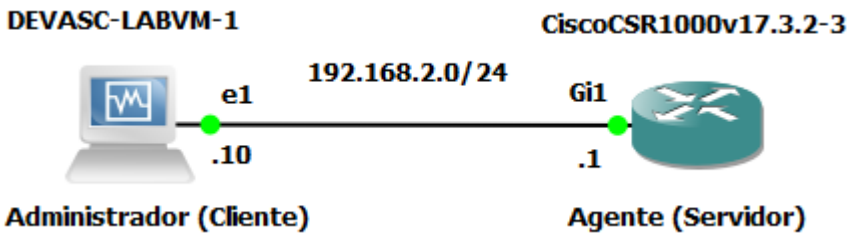


Figura 10. Topología de red

6.2. Explorar modelos YANG en GITHUB

a. Iniciar la máquina virtual DEVASC-LABVM

b. Abrir Chromium y dirigirse al siguiente enlace: https://github.com/YangModels/yang

c. En la página dirigirse a los modelos de datos YANG para Cisco IOS XE versión 17.3.1 dando clic en las siguientes carpetas: vendor → cisco → xe → 1731.

d. En la lista encuentre los modelos IETF y dar clic en el ietf-interfaces.yang. Examinar todo el modelo de datos e identificar, escoger y describir un nodo `leaf`, un `leaf-list`, un `container` y un `list`.

NOTA: Recuerde que el módulo es aquel que define un único modelo de datos. En este literal se explorará el modelo de datos "ietf-interface" que tiene asociado su identificación única por el XML namespace: "urn:ietf:params:xml:ns:yang:ietf-interfaces" (ver Figura 11). Dicho modelo es establecido por la organización IETF y muestra una organización jerárquica de los datos que se encuentran en las interfaces de los dispositivos.

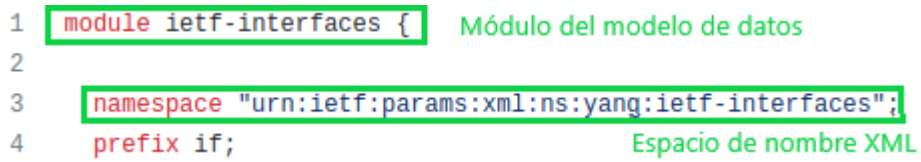


Figura 11. Parte del modelo de datos ietf-interfaces

e. En la lista encuentre los modelos OPENCONFIG y dar clic en openconfig-interfaces.yang. Examinar todo el modelo de datos e identificar, escoger y describir un nodo `leaf`, un `leaf-list`, un `container` y un `list`.

NOTA: En este literal se explorará un modelo de datos establecido por la organización OPENCONFIG. Al igual que el modelo de IETF se describe una organización jerárquica de los datos que contienen las interfaces de los dispositivos, sin embargo, la forma en cómo se organizan los datos es muy diferente.

Note que este modelo de datos tiene asociado el módulo "openconfig-interfaces" con su respectivo XML namespace: "http://openconfig.net/yang/interfaces" (ver Figura 12)

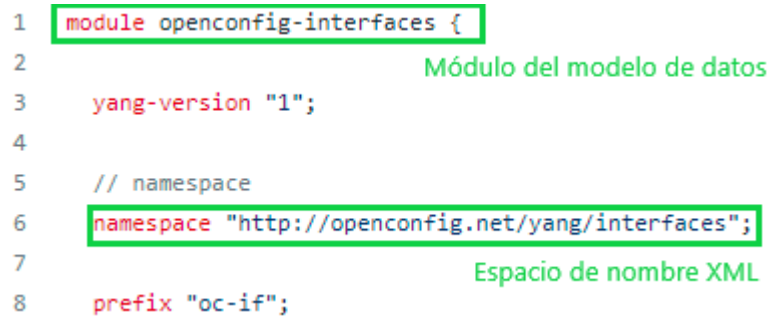


Figura 12. Parte del modelo de datos openconfig-interfaces

6.3. Explorar modelos YANG con pyang

a. Dirigirnos al directorio /labs/devnet-src/ y crear un subdirectorio llamado pyang.

```
devasc@labvm:~$ cd labs/devnet-src/
devasc@labvm:~/labs/devnet-src$ mkdir pyang
```

b. En este punto se debe tener abierto los modelos YANG del literal 6.1. Dirigirse al inicio de estos modelos, dar clic en la opción Raw (parte superior derecha) y copiar el URL. Luego emplear el comando `wget [URL]` para guardar los modelos en el subdirectorio pyang.

NOTA: Es necesario deshabilitar IPV6 en la máquina virtual DEVASC-LABVM para poder descargar los modelos YANG, para ello debe emplear los siguientes comandos:

```
sudo sysctl -w net.ipv6.conf.all.disable_ipv6=1
sudo sysctl -w net.ipv6.conf.default.disable_ipv6=1
sudo sysctl -w net.ipv6.conf.lo.disable_ipv6=1
```

c. Verificar si `pyang` está instalado, para ello emplear el comando `pyang -v`.

Obtendrá una salida parecida a la siguiente:

```
devasc@labvm:~/labs/devnet-src/pyang$ pyang -v
pyang 2.2.1
```

d. Actualizar `pyang` con el comando `pip3 install pyang --upgrade`.

e. Visualizar las opciones del comando `pyang`, para ello emplear el comando `pyang -h | more`. Prestar mayor atención en la opción -f ya que presenta los formatos para transformar los modelos YANG. (ver Figura 13)

```
                      Apply transform TRANSFORM.  Supported transforms are:
                      edit
-f FORMAT, --format=FORMAT
                      Convert to FORMAT.  Supported formats are: yang, yin,
                      dsdl, identifiers, tree, name, uml, depend, omni,
                      sample-xml-skeleton, capability, jsonxsl, jstree, jtox
-o OUTFILE, --output=OUTFILE
                      Write the output to OUTFILE instead of stdout.
```

Figura 13. Opción -f de pyang

f. Transformar los modelos de datos en formato de árbol. Seguramente notará que ambos modelos son mucho más fáciles de entender en su totalidad. Analizar y describir cada una de las salidas. (ver Figura 14 y Figura 15)

NOTA: Algunas cosas a tener en cuenta sobre la salida son las siguientes:

- (ro): Nodo solo para lectura

- (rw): Nodo para lectura y escritura
- (?): Nodo opcional

```
devasc@labvm:~/labs/devnet-src/pyang$ pyang -f tree ietf-
interfaces.yang
```

Figura 14. Modelo de datos ietf-interfaces en formato árbol

```
devasc@labvm:~/labs/devnet-src/pyang$ pyang -f tree openconfig-
interfaces.yang
```

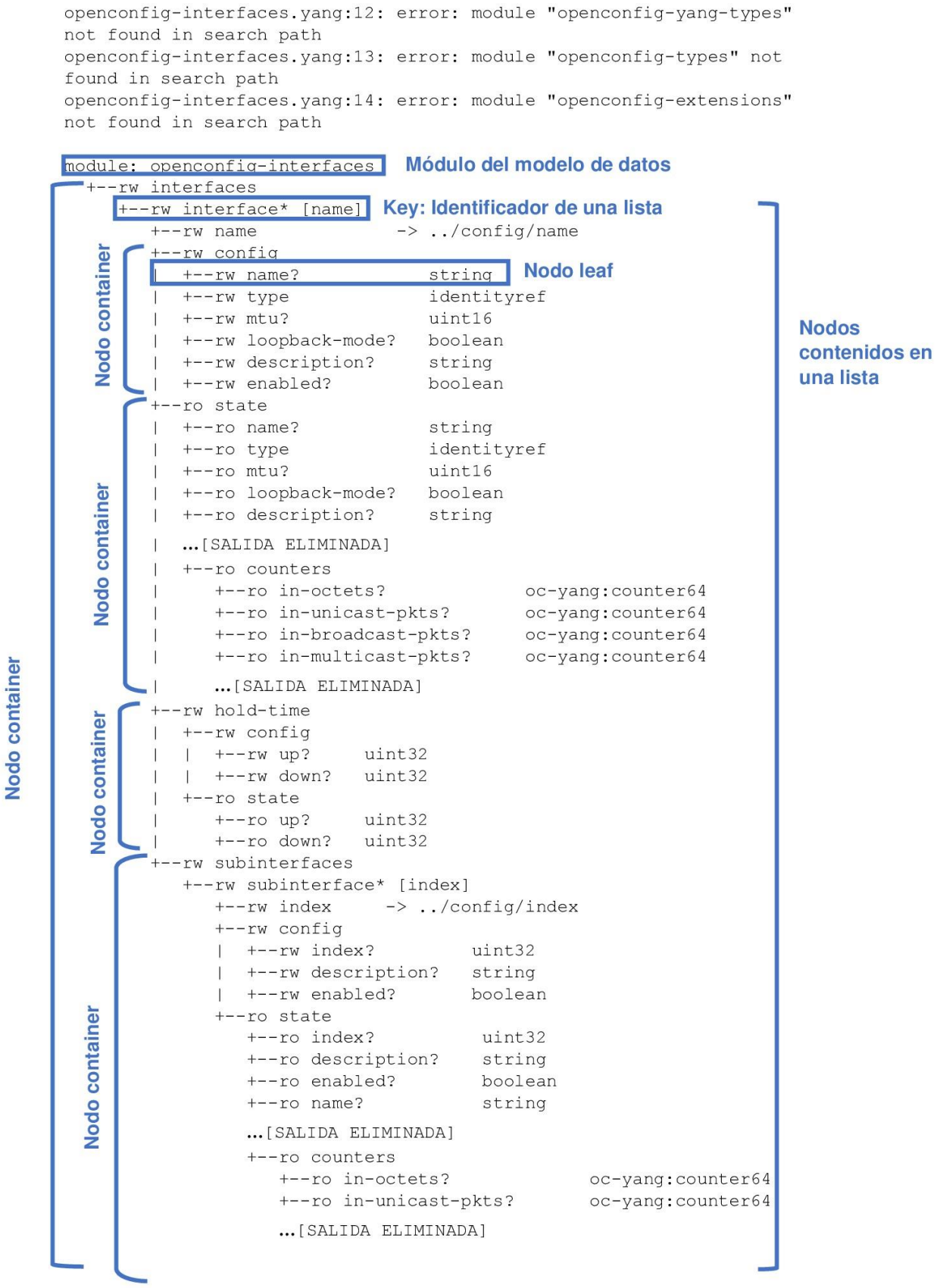


Figura 15. Modelo de datos openconfig-interfaces en formato árbol

6.4. Ejemplo de la representación de datos de un dispositivo de red con YANG

a. Descargar el archivo *example-ietf-interfaces-data.xml* del siguiente link. https://drive.google.com/drive/folders/1PpffXFFhZ53bHO47ZprtouNtB7o1oelz?usp=sharing https://drive.google.com/drive/folders/1PpffXFFhZ53bHO47ZprtouNtB7o1oelz

b. Abrir el archivo *example-ietf-interfaces-data.xml* y examinar el mismo. Este archivo es un ejemplo de los datos devueltos desde un dispositivo de red que muestra información sobre sus interfaces.

NOTA: Tome en cuenta que el ejemplo tiene formato XML, esto debido a que la sintaxis y la codificación que emplea NETCONF para enviar y recibir información es XML. Al examinar el ejemplo notará que la información mostrada en formato XML sigue la organización jerárquica del modelo de datos "ietf-interfaces". (ver Figura 16)

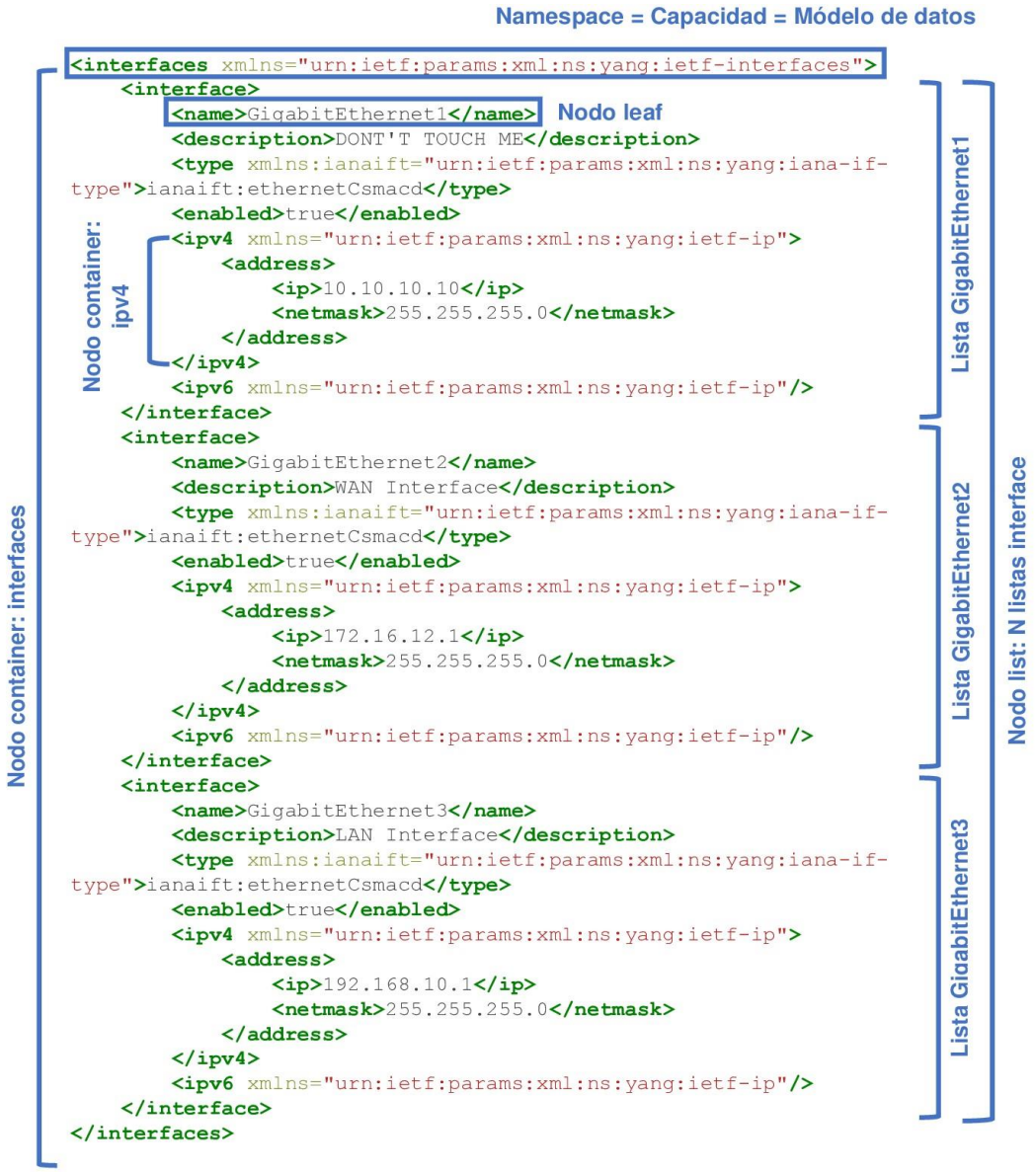


Figura 16. Ejemplo XML de datos devueltos por un dispositivo de red

c. Analizar y comparar el ejemplo del literal anterior con el modelo de datos “ietf-interfaces”. Comente lo descubierto.

d. Responder lo siguiente: ¿A qué se debe la aparición de los nodos `container` ipv4 e ipv6, si estos no fueron descritos en el modelo “ietf-interfaces”?

NOTA: Se recomienda examinar el modelo de datos “ietf-ip” específicamente la declaración `augment` que contine los nodos ipv4 e ipv6.

6.5. Identificar modelos de datos YANG que permite el Agente

a. Encender el router CSR 1000v (Agente).
b. Acceder al proceso de NETCONF a través de un subsistema de SSH con usuario admin2.

```
devasc@labvm:~$ ssh admin2@192.168.2.1 -p 830 -s netconf
```

c. Identificar el XML namespace: “urn:ietf:params:xml:ns:yang:ietf-interfaces” en el mensaje <hello> enviado por el Agente.

Recuerde que al iniciar el servicio NETCONF, el Agente es el primero en anunciar sus capacidades mediante un mensaje <hello>. Es ahí donde se verifica los modelos de datos YANG que soporta el Agente. Es importante recalcar que los modelos de datos no son capacidades, pero se anuncian como si lo fueran. El formato de los modelos de datos anunciados suele ser igual o parecida a del ejemplo mostrado a continuación.

```
<capability>urn:ietf:params:xml:ns:yang:ietf-
interfaces?module=ietf-interfaces&amp;revision=2014-05-
08&amp;features=pre-provisioning,if-mib,arbitrary-
names&amp;deviations=cisco-xe-ietf-ip-deviation</capability>
```

Es necesario segmentar dicho ejemplo para poder tener una mejor apreciación del formato que manejan este y otros modelos de datos que son anunciados como capacidades por el Agente. A continuación, se describe en orden el formato de dicho ejemplo.

1. **Namespace-URI:** `urn:ietf:params:xml:ns:yang:ietf-interfaces`
2. **Carácter “?”**
3. **Nombre del modelo de datos:** `module=ietf-interfaces`
4. **Carácter “&”**
5. **Fecha de versión del modelo de datos:** `revision=2014-05-08`
6. **Carácter “&”**
7. **Funciones condicionales establecidas en el modelo de datos:**
   `features=pre-provisioning,if-mib,arbitrary-names`
8. **Carácter “&”**
9. **Desviación - Otro modelo que modifica al modelo de datos:**
   `deviations=cisco-xe-ietf-ip-deviation`

NOTA: La secuencia de caracteres `&` es usada para representar el carácter “&”

d. Iniciar sesión NETCONF enviando mensaje <hello>. Para ello pegar el siguiente texto XML en la terminal.

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.0</capability>
  </capabilities>
</hello>]]>]]>
```

e. Solicitar información de las interfaces del Agente mediante la operación <get>. Para ello pegar el siguiente texto en la terminal.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="103">
  <get>
    <filter>
      <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-
interfaces"/>
    </filter>
  </get>
</rpc>]]>]]>
```

NOTA: Tome en cuenta que el elemento <interfaces> es parte del modelo de datos "ietf-interfaces" que identificó en el mensaje <hello> que recibió de parte del Agente. Es necesario declarar el XML namespace de dicho modelo de datos como un atributo en el elemento <interfaces>, caso contrario se ocasionará un error ya que el elemento no tendrá ninguna referencia del modelo de datos.

f. Analizar la respuesta obtenida por parte del Agente. Emplear el siguiente link: https://www.samltool.com/prettyprint.php para obtener una mejor apreciación de la información.

g. Cerrar sesión NETCONF. Para ello pegar el siguiente texto en la terminal

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-
id="1239123">
  <close-session/>
</rpc>
```

7. INFORME

7.1. Presentar capturas de pantalla del procedimiento realizado en la práctica con la debida explicación de ser necesario.
NOTA: Algunos enunciados en el procedimiento, requieren respuesta o solicitan algún análisis. Recuerde incluir los mismos.
7.2. Detallar las diferencias entre los modelos de datos YANG vistos en el laboratorio.
7.3. Consultar y describir las declaraciones del lenguaje YANG: `typedef`, `grouping`, `uses`, `choices` y `rpc`.
7.4. Conclusiones y Recomendaciones.

8. REFERENCIAS

Las referencias de la práctica 3 se encuentran en el apartado de referencias de este documento.

PRÁCTICA N° 4

1. TEMA

OPERACIONES Y CAPACIDADES NETCONF

2. OBJETIVOS

2.1. Utilizar mensajes <rpc> y <rpc-reply> para el intercambio de información entre Administrador y Agente.
2.2. Explicar el funcionamiento de las operaciones base de NETCONF.
2.3. Comprender que son las capacidades de NETCONF.

3. MARCO TEÓRICO

3.1. Modelo RPC

NETCONF emplea un modelo de comunicación basado en RPC, en donde los pares (cliente y servidor) usan elementos <rpc> y <rpc-reply> para el intercambio de mensajes de solicitud y respuesta respectivamente [7].

Elemento <rpc>

Se emplea para encerrar una solicitud NETCONF que es enviada desde el cliente al servidor. El elemento <rpc> contine un atributo obligatorio del tipo string llamado “message-id” que es elegido por el remitente de la RPC, la cual va incrementando en una unidad. Dicho string, así como cualquier otro atributo adicional presente en el elemento <rpc> deberán ser devueltos sin ninguna modificación en el elemento <rpc-reply> [7].

Elemento <rpc-reply>

Se envía como repuesta al mensaje <rpc>. El elemento <rpc-reply> contiene el atributo obligatorio "message-id, que es igual al atributo "message-id" del mensaje <rpc> [7].

Un elemento <rpc-replay> puede contener una jerarquización de datos que fue solicitada por parte del cliente, así también puede contener las siguientes estructuras <rpc>:

1. Elemento <rpc-error>: Se envía en el caso de que ocurra un error en el proceso de solicitud de <rpc> [7].

2. Elemento <ok>: Se envía si no existió errores o alarmas durante el proceso de solicitud de <rpc> y cuando no se retorna datos de una solicitud realizada [7].

3.2. Almacenes de datos de configuración

Un almacén de datos de configuración se define como un conjunto completo de datos de configuración para que un dispositivo de red pase de su estado predeterminado inicial a un estado operativo deseado [7]. La Tabla 1 muestra una descripción de algunos almacenes de datos de configuración [7].

Tabla 1. Almacenes de datos de configuración

Almacén de datos de configuración	**Descripción**

<running>	Almacén de datos de configuración predeterminado que siempre está presente en un dispositivo de red. Contiene la configuración completa que actualmente está activa en el mismo.
<candidate>	Depende de la capacidad Candidate y se usa comúnmente para almacenar los datos de configuración que se pueden manipular sin afectar la configuración actual del dispositivo.
<startup>	Depende de la capacidad Startup y es aquel que es cargado por el dispositivo como parte de su inicialización cuando se reinicia o se recarga.

3.3. Operaciones base

NETCONF proporciona un pequeño conjunto de operaciones de bajo nivel para administrar configuraciones de dispositivos y recuperar información de estado de dispositivos [7]. La Tabla 2 muestra una pequeña descripción de dichas operaciones.

Tabla 2. Operaciones base de NETCONF

Operación base	**Descripción**
<get>	Recupera información de configuración y de estado solo del almacén de datos <running>.
<get-config>	Recupera información de configuración de todo o parte de un almacén de datos especifico.
<edit-config>	Edita toda o parte de una configuración de un almacén de datos especifico.
<copy-config>	Copia el contenido de un almacén de datos de configuración en otro.
<delete-config>	Elimina un almacén de datos de configuración.
<close-session>	Cierra una sesión NETCONF correctamente.
<lock>/<unlock>	Bloquea y desbloquea un almacén de datos de configuración.
<kill-session>	Fuerza el cierra de sesión NETCONF.

3.4. Capacidades

Una capacidad de NETCONF es un conjunto de funciones extra que aumentan las especificaciones base del protocolo. Un cliente puede hacer uso de las capacidades de un servidor, si estas fueron anunciadas por el mismo en el intercambio inicial de mensajes <hello> [7]. Algunas capacidades son descritas en la Tabla 3.

Tabla 3. Capacidades NETCONF

Capacidad	**Descripción**
Base	Es la capacidad que definen todos los elementos del protocolo NETCONF. Existen dos versiones la 1.0 y 1.1(actual) [7].

Writable-running	Permite realizar cambios de configuración directos en el almacén de datos <running> [7].
Rollback-on-error	Indica que el dispositivo admite el valor "rollback-on-error" en el parámetro <error-option> de la operación <edit-config> [7].
Validate	Permite comprobar si una configuración completa no posee errores sintácticos y sistemáticos antes de aplicar la configuración en el dispositivo [7].
Xpath	Indica que el dispositivo admite el uso de expresiones XPath en el elemento <filter> [7].
Notification	Permite al dispositivo recibir, procesar y responder solicitudes NETCONF mientras una suscripción de notificación está activa [26].
Interleave	Permite al dispositivo recibir, procesar y responder solicitudes NETCONF mientras una suscripción de notificación está activa [26].
With-defaults	Permite al cliente conocer como el dispositivo maneja los valores predeterminados. Además, define nuevos mecanismos para que el cliente pueda controlar los valores predeterminados del dispositivo [27].
Yang-library	Informa al cliente que el router implementa la biblioteca de módulos YANG [10].
Actions	Introduce una nueva operación RPC que permite invocar acciones (metodos) definidas en un modelo de datos. Esta capacidad está obsoleta, ya que las acciones ahora se admiten en el estándar YANG 1.1 [12].

4. TRABAJO PREPARATORIO

4.1. Revisar el marco teórico para la realización de la práctica.
4.2. Consultar los valores del atributo “operation” disponibles en la operación base <edit-config>.
4.3. Consultar los valores de que pueden tomar <test-option> y <error-option> en la operación base <edit-config>.
4.4. Consultar qué información se muestra en el elemento <rpc-error>.
4.5. Consultar sobre la operación <commit>.

5. EQUIPO Y MATERIALES

- PC (GNS3)

6. PROCEDIMIENTO

6.1. Topología de red

a. Implementar la topología de red mostrada en la Figura 1.
NOTA: Emplear el proyecto de GNS3 creado en la práctica 2.

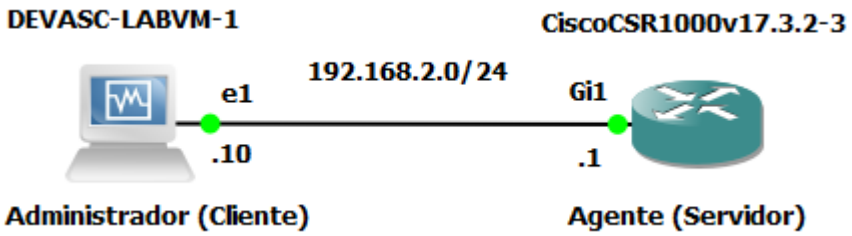


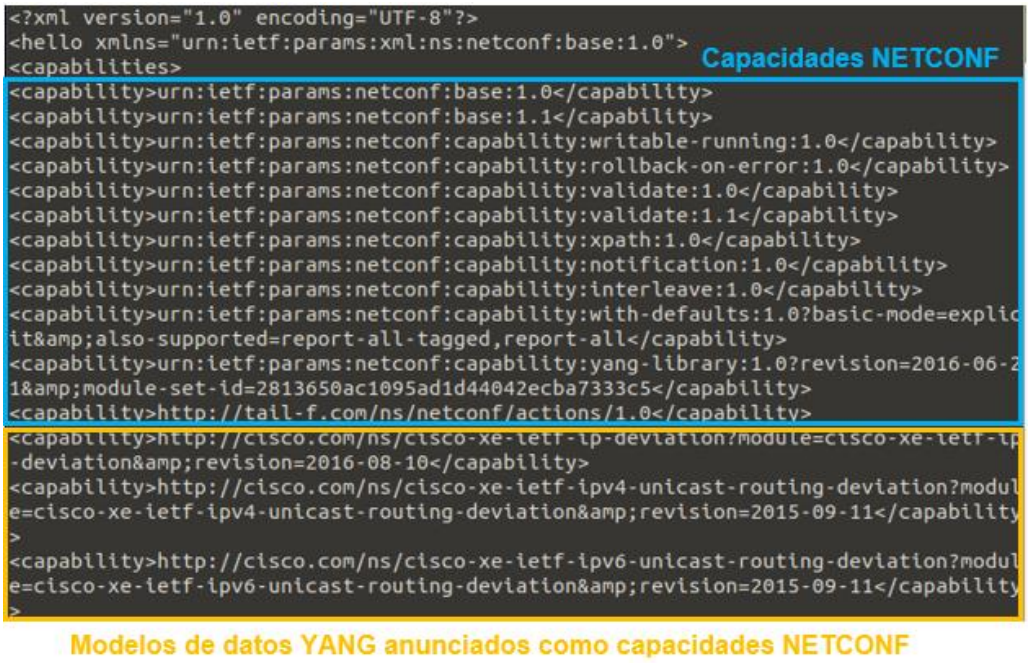
Figura 1. Topología de red

6.2. Explorar las capacidades del Agente

a. Acceder al proceso NETCONF a través de un subsistema de SSH con usuario admin2.

```
devasc@labvm:~$ ssh admin2@192.168.2.1 -p 830 -s netconf
```

b. Identificar las capacidades del Agente, las cuales están contenidas en el mensaje <hello>.




```
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<capabilities>
<capability>urn:ietf:params:netconf:base:1.0</capability>
<capability>urn:ietf:params:netconf:base:1.1</capability>
<capability>urn:ietf:params:netconf:capability:writable-running:1.0</capability>
<capability>urn:ietf:params:netconf:capability:rollback-on-error:1.0</capability>
<capability>urn:ietf:params:netconf:capability:validate:1.0</capability>
<capability>urn:ietf:params:netconf:capability:validate:1.1</capability>
<capability>urn:ietf:params:netconf:capability:xpath:1.0</capability>
<capability>urn:ietf:params:netconf:capability:notification:1.0</capability>
<capability>urn:ietf:params:netconf:capability:interleave:1.0</capability>
<capability>urn:ietf:params:netconf:capability:with-defaults:1.0?basic-mode=explic
it&amp;also-supported=report-all-tagged,report-all</capability>
<capability>urn:ietf:params:netconf:capability:yang-library:1.0?revision=2016-06-2
1&amp;module-set-id=2813650ac1095ad1d44042ecba7333c5</capability>
<capability>http://tail-f.com/ns/netconf/actions/1.0</capability>
<capability>http://cisco.com/ns/cisco-xe-ietf-ip-deviation?module=cisco-xe-ietf-ip
-deviation&amp;revision=2016-08-10</capability>
<capability>http://cisco.com/ns/cisco-xe-ietf-ipv4-unicast-routing-deviation?modul
e=cisco-xe-ietf-ipv4-unicast-routing-deviation&amp;revision=2015-09-11</capability
>
<capability>http://cisco.com/ns/cisco-xe-ietf-ipv6-unicast-routing-deviation?modul
e=cisco-xe-ietf-ipv6-unicast-routing-deviation&amp;revision=2015-09-11</capability
>
```

Figura 2. Parte del mensaje <hello> enviado por el Agente

Para poder establecer una sesión NETCONF entre Administrador y Agente es necesario realizar un intercambio de mensajes <hello>. Recuerde que el primero en anunciar sus capacidades mediante un mensaje <hello> es el Agente. De la larga lista de capacidades que anuncia dicho dispositivo se puede identificar dos grupos: Las capacidades NETCONF y los modelos de datos YANG anunciados como capacidades NETCONF (ver Figura 2). Recuerde que a pesar que los modelos de datos no son netamente capacidades, se las anuncia como si las fueran. Se suele reconocer a las mismas ya que poseen el parámetro “module” que es nombre del módulo al que pertenece el modelo de datos.

c. Leer el siguiente análisis sobre las capacidades NETCONF.

Note que el dispositivo posee algunas capacidades NETCONF. En la Tabla 4 se muestran las mismas con su respectiva referencia [28]. Si desea aprender más sobre estas capacidades puede consultar los RFCs mostrados en dicha tabla.

Tabla 4. Capacidades NETCONF disponibles en el Agente

Capacidad	URI-Identificador	Referencia
Base	urn:ietf:params:netconf:base:1.0	RFC 4741
	urn:ietf:params:netconf:base:1.1	RFC 6241
Writable-running	urn:ietf:params:netconf:capability:writable-running:1.0	
Rollback-on-error	urn:ietf:params:netconf:capability:writable-running:1.0	
Validate	urn:ietf:params:netconf:capability:validate:1.0	RFC 4741
	urn:ietf:params:netconf:capability:validate:1.1	RFC 6241
Xpath	urn:ietf:params:netconf:capability:xpath:1.0	
Notification	urn:ietf:params:netconf:capability:notification:1.0	RFC 5277
Interleave	urn:ietf:params:netconf:capability:interleave:1.0	
With-defaults	urn:ietf:params:netconf:capability:with-defaults:1.0	RFC 6243
Yang-library	urn:ietf:params:netconf:capability:yang-library:1.0	RFC 7950
Actions*	http://tail-f.com/ns/netconf/actions/1.0	-

*Capacidad obsoleta, las acciones se definen en el RFC 7950.

De las capacidades anunciados por el Agente se concluye que el único almacén de datos disponible en el mismo es el almacén <running>. Debido a que el dispositivo no anunció la capacidad Startup y Candidate, no puede hacer uso de los almacenes de datos <startup> y <candidate>.

d. Terminar el proceso NETCONF ejecutado en el literal b.
e. Habilitar el almacén de datos <candidate> en el Agente.

```
CSR1k(config)#netconf-yang feature candidate-datastore
CSR1k(config)#do wr
```

NOTA: Al habilitar el almacén de datos <candidate>**,** todo el contenido del almacén de datos de configuración <running> será copiado al nuevo almacén de datos <candidate>. Tome en cuenta que <running> tiene activa una configuración.

f. Volver a iniciar el proceso NETCONF a través de un subsistema de SSH con usuario admin2 y de nuevo identificar las capacidades del Agente, que están contenidas en el mensaje <hello>.

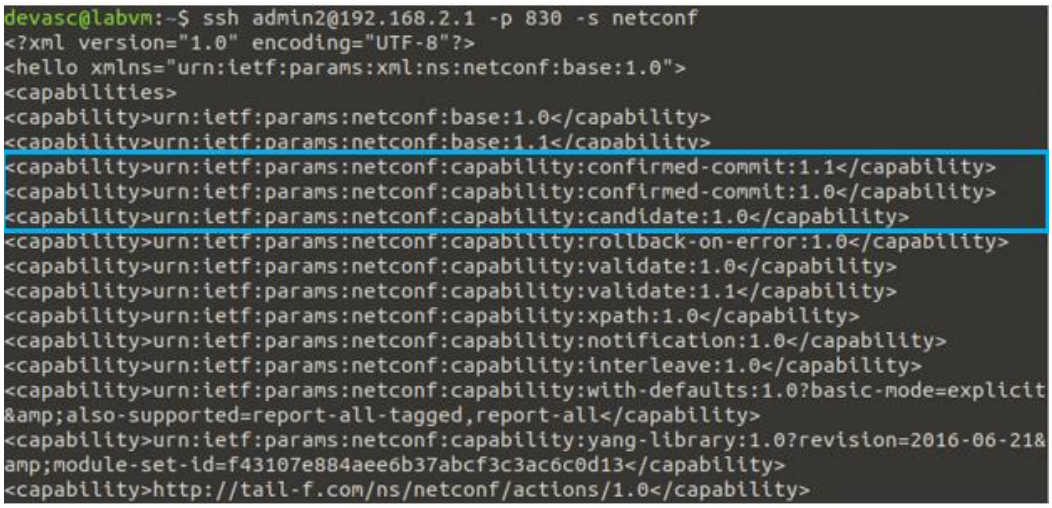


Figura 3. Parte del mensaje <hello> enviado por el Agente luego de habilitar el almacén de datos de configuración <candidate>

Al habilitar el almacén de datos de configuración <candidate> en el dispositivo, se obtienen las siguientes consideraciones con respecto a las capacidades anunciadas por el Agente (ver Figura 3):

- Desaparece la capacidad Writable-running.
- Aparecen dos nuevas capacidades. La capacidad Candidate y la capacidad Confirmed-commit. La primera capacidad indica que el dispositivo admite el almacén de datos <candidate> e introduce dos nuevas operaciones: <commit> y <discard-changes>. Mientras que la segunda capacidad indica que el dispositivo admite una nueva operación <cancel-commit>.

g. Iniciar sesión NETCONF enviando mensaje <hello> al Agente. Pegar el siguiente texto XML en la terminal.

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.0</capability>
  </capabilities>
</hello>]]>]]>
```

6.3. Enviar mensajes RPC al Agente empleando las operaciones base de NETCONF

a. Leer el siguiente análisis sobre los mensajes RPC.
Los elementos <rpc> y <rpc-reply> son empleados para el intercambio de mensajes de solicitud y respuesta respectivamente. Para poder solicitar al Agente realizar cualquier operación es necesario enviar un mensaje <rpc> que contenga dicha operación. La estructura de dicho mensaje es la siguiente:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="XXX">
  <..Operación NETCONF..>
</rpc>
```

Al terminar de ejecutar la operación solicitada, el Agente enviará un mensaje <rpc-reply> que contendrá ya sea la información requerida, un elemento de operación exitosa <ok> o un elemento <rpc-error> notificando algún tipo de error. La estructura de este mensaje se muestra a continuación:

```
< rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-
id="XXX">
  <..Información devuelta..> / <ok> / <rpc-eror>
</rpc-reply>
```

NOTA: Para los siguientes literales se empleará el modelo de datos: ietf-interfaces. Recuerde que se puede emplear cualquier modelo de datos anunciado como capacidad en el Agente. Para su uso es necesario tener un conocimiento previo del modelo de datos, ya que esto le permitirá seleccionar aquel que mejor se adapte a sus necesidades.

b. Emplear la operación <get> para obtener la información de la interfaz Gigabit Ethernet 1 del router. Para ello pegar el siguiente texto XML en la terminal.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="103">
  <get>
    <filter>
      <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
        <interface>
          <name>GigabitEthernet1</name>
        </interface>
      </interfaces>
      <interfaces-state xmlns="urn:ietf:params:xml:ns:yang:ietf-
interfaces">
        <interface>
          <name>GigabitEthernet1</name>
        </interface>
      </interfaces-state>
    </filter>
  </get>
</rpc>]]>]]>
```

La operación <get> es empleada solo en el almacén de datos <running>. Lo que implica que permitirá extraer solo la información de configuración y de estado contenida en este almacén de datos. Note en el ejemplo, que la operación <get> contiene elemento <filter> el cual permite extraer todo o parte del almacén de datos <running>. Debido a que se requiere recuperar la información de la interfaz Gigabit Ethernet 1 del router, <filter> debe contener los nodos <interfaces> y <interfaces-state> del modelo de datos: ietf-interfaces.

c. Analizar el mensaje de respuesta del Agente <rpc-reply> después de solicitar la operación <get>.

d. Emplear la operación <get-config> para obtener la información de la interfaz Gigabit Ethernet 1 del router contenida en el almacén de datos <candidate>. Para ello pegar el siguiente texto XML en la terminal.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="104">
  <get-config>
    <source>
      <candidate/>
    </source>
```

```
    <filter>
      <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
        <interface>
          <name>GigabitEthernet1</name>
        </interface>
      </interfaces>
      <interfaces-state xmlns="urn:ietf:params:xml:ns:yang:ietf-
interfaces">
        <interface>
          <name>GigabitEthernet1</name>
        </interface>
      </interfaces-state>
    </filter>
  </get-config>
</rpc>]]>]]>
```

La operación <get-config> permite recuperar solo la información de configuración de cualquier almacén de datos permitido por el dispositivo. Note que en el ejemplo <get-config> contiene el elemento <source> el cual indica el almacén de datos al que consulta la operación, que en este caso es el almacén de datos <candidate>.

e. Analizar el mensaje de respuesta del Agente <rpc-reply> después de solicitar la operación <get-config>.

f. Emplear la operación <edit-config> para configurar la interfaz Gigabit Ethernet 3 del almacén de datos <candidate>. Para ello pegar el siguiente texto XML en la terminal.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="105">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
      <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
        <interface>
          <name>GigabitEthernet3</name>
          <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-
type">ianaift:ethernetCsmacd</type>
          <enabled>true</enabled>
          <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
            <address>
              <ip>192.168.4.1</ip>
              <netmask>255.255.255.0</netmask>
            </address>
          </ipv4>
        </interface>
      </interfaces>
    </config>
  </edit-config>
</rpc>]]>]]>
```

La operación <edit-config > es usada para cargar todo o una parte de una configuración en un determinado almacén de datos. <edit-config> suele tener cinco elementos <target>, <config>, <default-operation>, <test-option> y <error-option>, siendo los tres primeros los más fundamentales.

- El elemento <target> indica el almacén de datos que será editado. Note en el ejemplo, que <target> contiene el almacén de datos <candidate>.

- El elemento <config> contiene parte de la organización jerárquica de los datos de configuración que se establecen en un modelo de datos que emplea el dispositivo. En este elemento se suele usar el atributo "opertion", el cual puede aparecer en cualquier elemento del subárbol <config>. Dicho atributo posee cinco distintos valores que permiten modificar la forma de configuración de los datos. Debido a que dicho atributo es parte de XML namespace de NETCONF: *urn:ietf:params:xml:ns:netconf:base:1.0*, es necesario declarar el mismo cada vez que se haga uso del atributo. Esto resulta incómodo, razón por la cual se declara el namespace con un prefijo al inicio del elemento <config>. Note que en el ejemplo el prefijo asociado al namespace es "xc". Así también observe que el atributo "operation" no es empleado. No era necesario que <config> contenga namespace sin embargo se lo realizó debido a que es una buena práctica siempre declararlo. Por último, es importante mencionar que el ejemplo emplea el modo de operación "marge", que es el comportamiento predeterminado de <edit-config>. Tome en cuenta que no es necesario declarar el valor "marge" para el atributo "operation".

- El elemento <default-operation> es opcional, pero si es empleado puede tomar los valores "marge", "replace" o "none". Se suele omitir si se emplea el modo de operación "marge". Note que el ejemplo no se incluye el elemento <default-operation> ya que se emplea el dicho modo.

- Los elementos <test-option> y <error-option> son considerados opcionales. Si bien <test-option> ofrece distintos valores para verificar la existencia de errores y <error-option> ofrece valores que determinan la acción a tomar en el caso de la existencia de errores, ambos elementos no influyen de manera relevante en el modo de operación de <edit-config>.

g. Determinar porque no se incluyó los elementos <test-option> y <error-option> en el ejemplo del literal anterior. Recuerde que no incluir algo, no siempre significa que no esté funcionando.

h. Analizar el mensaje de respuesta del Agente <rpc-reply> después de solicitar la operación <edit-config>.

i. Emplear la operación <copy-config> para copiar la configuración del almacén de datos <candidate> al almacén de datos < running >. Para ello pegar el siguiente texto XML en la terminal.

```
<?xml version="1.0"?>
```

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="106">
  <copy-config>
    <target>
      <running/>
    </target>
    <source>
      <candidate/>
    </source>
  </copy-config>
</rpc>]]>]]>
```

La operación <copy-config> permite remplazar un almacén de datos de configuración completo con el contenido de otro almacén de datos de configuración. Dicha operación contiene los elementos <target> y <source>. El primero indica el almacén de datos de configuración que se usará como destino por la operación y el segundo indica el almacén de datos que se usará como fuente de la operación. Note que el ejemplo <target> corresponde al almacén de datos <running> y <source> al almacén de datos <candidate>.

j. Analizar el mensaje de respuesta del Agente <rpc-reply> después de solicitar la operación <copy-config> y luego contestar la siguiente pregunta: ¿Fue posible copiar la configuración del almacén de datos <candidate> al almacén de datos <running>? Justifique su respuesta.

k. Emplear la operación <delete-config> para eliminar la configuración del almacén de datos <running>. Para ello pegar el siguiente texto en la terminal.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="107">
  <delete-config>
    <target>
      <running/>
    </target>
  </delete-config>
</rpc>]]>]]>
```

La operación <delete-config> permite eliminar un almacén de datos de configuración. La operación contiene el elemento <target> que indica el almacén de datos de configuración que será eliminado. Note en el ejemplo, que <target> corresponde al almacén de datos <running>.

l. Analizar el mensaje de respuesta del Agente <rpc-reply> después de solicitar la operación <delete-config> y luego contestar la siguiente pregunta: ¿Fue posible eliminar la configuración del almacén de datos <running>? Justifique su respuesta.

m. Limpiar la configuración del almacén de datos <candidate>. Para ello pegar el siguiente texto XML en la terminal.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="108">
  <discard-changes/>
</rpc>]]>]]>
```

NOTA: En la sección 6.4 se tratará sobre la operación <discard-changes>

n. Emplear la operación <lock> para bloquear la configuración del almacén de datos <running> y <candidate>. Para ello pegar los siguientes textos XML en la terminal.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="109">
  <lock>
    <target>
      <running/>
    </target>
  </lock>
</rpc>]]>]]>

<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="110">
  <lock>
    <target>
      <candidate/>
    </target>
  </lock>
</rpc>]]>]]>
```

La operación <lock> permite al Administrador bloquear todo un almacén de datos de configuración, dándole así seguridad de poder realizar cambios en la configuración sin que otros clientes intervengan. El bloqueo dura hasta que se cierra una sesión NETCONF o se libera el bloqueo. La operación contiene el elemento <target> que indica el almacén de datos a bloquear, note que en los ejemplos corresponde a los almacenes <running> y <candidate>.

o. Analizar los mensajes de respuesta del Agente <rpc-reply> después de solicitar la operación <lock>.

p. Iniciar sesión NETCONF en otra terminal y en la misma intentar configurar la interfaz Gigabit Ethernet 3 del almacén de datos <candidate>. También ingresar a la CLI del router e intentar realizar la misma configuración. Tenga en cuenta que al usar el CLI está configurando sobre el almacén de datos <running>. Comente los resultados.

NOTA: Primero emplear el comando `show netconf-yang sessions` para verificar las sesiones NETCONF y luego utilizar el siguiente texto XML para intentar configurar la interfaz en el almacén de datos de configuración <candidate>.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="103">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
      <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
        <interface>
```

```
          <name>GigabitEthernet3</name>
          <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-
type">ianaift:ethernetCsmacd</type>
          <enabled>true</enabled>
          <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
            <address>
              <ip>192.168.4.1</ip>
              <netmask>255.255.255.0</netmask>
            </address>
          </ipv4>
        </interface>
      </interfaces>
    </config>
  </edit-config>
</rpc>]]>]]>
```

q. Emplear la operación <unlock> para desbloquear la configuración del almacén de datos <running> y <candidate>. Recuerde emplear dicha operación en la sesión que utilizó para bloquear los almacenes de datos. Pegue los siguientes textos XML en esa terminal.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="111">
  <unlock>
    <target>
      <running/>
    </target>
  </unlock>
</rpc>]]>]]>
```

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="112">
  <unlock>
    <target>
      <candidate/>
    </target>
  </unlock>
</rpc>]]>]]>
```

La operación <unlock> permite liberar el bloqueo del almacén de datos de configuración. El desbloqueo será exitoso si el bloqueo está actualmente activo y si la sesión que emitió el bloqueo es la misma que intenta desbloquear. La operación contiene <target> que indica el almacén de datos a desbloquear. Note que en los ejemplos corresponde a los almacenes de datos de configuración <running> y <candidate>.

r. Analizar el mensaje de respuesta del Agente <rpc-reply> después de solicitar la operación <unlock> y repita el literal n.

NOTA: Emplee la misma sesión NETCONF establecida en el literal n.

s. Emplear la operación <close-session> para terminar las sesiones NETCONF. Para ello pegar el siguiente texto XML en las terminales.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="113">
  <close-session/>
</rpc>
```

t. Es también posible emplear la operación <kill-session> para forzar la terminación de la sesión NETCONF. Note el siguiente ejemplo.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="114">
  <kill-session>
    <session-id>30</session-id>
  </kill-session>
</rpc>]]>]]>
```

La operación <kill-session> posee el elemento <session-id> que es el identificador de sesión NETCONF que va a ser forzado a terminar.

u. Iniciar sesión NETCONF en dos terminales distintas y luego usar el comando <kill-session> en la primera terminal para forzar la terminación de la sesión NETCONF de la segunda terminal. ¿Fue posible? Justique su respuesta.

NOTA: Verificar las sesiones NETCONF con el comando `show netconf-yang sessions` antes y después de emplear <kill-session>.

v. Dejar abierta una sesion NETCONF para la seguiente parte.

6.4. Operaciones adicionales permitidas por las capacidades anunciadas en el Agente

Las capacidades son un conjunto de funcionalidades que permiten incrementar las especificaciones base de NETCONF. Por ejemplo, algunas capacidades mejoran el funcionamiento de las operaciones base y otras introducen nuevas operaciones que ayudan a mejorar el funcionamiento de NETCONF.

NOTA: Los siguientes literales muestran algunas nuevas funcionalidades de NETCONF que son permitidas gracias a las capacidades anunciadas por el Agente.

Capacidad Candidate

a. Verificar que la interfaz Gigabit Ethernet 3 del almacén de datos <candidate> se encuentre configura con la dirección 192.168.4.1/24.
b. Emplear la CLI para verificar que la interfaz Gigabit Ethernet 3 no tenga asociada ninguna dirección IP. Caso contrario eliminar esta configuración de la interfaz.
c. Emplear la operación <commit>. Para ello pegar el siguiente texto XML en la terminal.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="115">
  <commit/>
</rpc>]]>]]>
```

d. Añadir una PC a la topología, configurar la dirección 192.168.4.10/24 y conectar la misma al puerto Gigabit Ethernet 3 del router. ¿Se logró establecer la conexión? Justifique su respuesta.

NOTA: Realizar prueba ping.

e. Emplear la operación <discard-changes>. Para ello pegar el siguiente texto XML en la terminal.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="116">
  <discard-changes/>
</rpc>]]>]]>
```

La operación <discard-changes> es empleada en el caso de que ya no se requiera usar la operación <commit>. Por ejemplo, imagine que usted ha cometido una equivocación grave en la configuración del almacén de datos <candidate>. Ya no sería factible usar <commit> y necesariamente le tocará volver a realizar las configuraciones. Ahí es cuando sea emplea <discard-changes>. Dicha operación sirve para deshacer los cambios de configuración realizados en el almacén de datos <candidate> y que el contenido de <candidate> coincida con el contenido de <running>.

f. Analizar el mensaje de respuesta del Agente <rpc-reply> después de solicitar la operación <discard-changes > y repita el literal d.

Capacidad Confirmed-commit:1.1

Dicha capacidad modifica la operación <commit> ya que permite implementar cuatro parámetros: <confirmed>, <confirmed-timeout>, <persist> y <persist-id>.

g. Emplear la operación <edit-config> para configurar la interfaz Gigabit Ethernet 4 del almacén de datos <candidate>. Para ello pegue el siguiente texto XML en la terminal.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="117">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
      <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
        <interface>
          <name>GigabitEthernet4</name>
          <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-
type">ianaift:ethernetCsmacd</type>
          <enabled>true</enabled>
          <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
            <address>
              <ip>192.168.5.1</ip>
              <netmask>255.255.255.0</netmask>
            </address>
          </ipv4>
        </interface>
      </interfaces>
    </config>
  </edit-config>
</rpc>]]>]]>
```

h. Conectar la PC a la interfaz Gigabit Ethernet 4 y modificar la dirección IP a la 192.168.5.10/24.

i. Emplear la operación <commit> y habilitar la confirmación de <commit> para un tiempo de 2 minutos. Para ello pegar el siguiente texto XML en la terminal.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="118">
  <commit>
    <confirmed/>
    <confirm-timeout>120</confirm-timeout>
  </commit>
</rpc>]]>]]>
```

NOTA: <confirm-timeout> define un periodo de tiempo límite en segundos para volver a confirmar <commit>.

j. Verificar la conectividad entre PC y router, y luego de dos minutos intentar realizar la misma prueba de conectividad. Comente los resultados y luego responda ¿Qué significa confirmar <commit>?

k. Emplear la operación <commit> y habilitar la confirmación de <commit> pero esta vez con un tiempo de 5 minutos. Emplear el siguiente texto XML.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="119">
  <commit>
    <confirmed/>
    <confirm-timeout>300</confirm-timeout>
  </commit>
</rpc>]]>]]>
```

l. Verificar la conectividad entre PC y router.

m. Emplear la operación <cancel-commit> para cancelar la operación <commit> que se encuentra en curso. Luego volver realizar la prueba de conectividad. Comente los resultados.

El texto de la operación <cancel-commit> que debe pegar en la terminal es el siguiente.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="120">
  <cancel-commit/>
</rpc>]]>]]>
```

NOTA: <cancel-commit> es usada solo en operaciones <commit> que necesita confirmación.

Capacidad Validate

n. Emplear la operación <validate> para verificar el contenido en el almacén de datos <candidate>. Para ello pegar el siguiente texto en la terminal.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="121">
  <validate>
```

```
    <source>
      <candidate/>
    </source>
  </validate>
</rpc>]]>]]>
```

o. Analizar el mensaje de respuesta del Agente <rpc-reply> después de solicitar la operación <validate>.

p. Cerrar sesión NETCONF.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="122">
  <close-session/>
</rpc>]]>]]>
```

7. INFORME

7.1. Presentar capturas de pantalla del procedimiento realizado en la práctica con la debida explicación de ser necesario.
NOTA: Algunos enunciados en el procedimiento, requieren respuesta o solicitan algún análisis. Recuerde incluir los mismos.

7.2. Emplear la operación base <edit-config> para eliminar la interfaz Gigabit Ethernet 4 del almacén de datos <candidate> que fue creada en el desarrollo de la práctica.
NOTA: Recuerde que el atributo “operation” con el valor “delete”, permite eliminar parte de la configuración.

7.3. Conclusiones y Recomendaciones.

8. REFERENCIAS

Las referencias de la práctica 4 se encuentran en el apartado de referencias de este documento.

PRÁCTICA N° 5

1. TEMA

CLIENTE NETCONF CON PYTHON

2. OBJETIVOS

2.1. Establecer una sesión NETCONF entre Administrador y Agente empleando ncclient.
2.2. Utilizar ncclient para recuperar información del Agente.
2.3. Utilizar ncclient para realizar cambios en la configuración del Agente.

3. MARCO TEÓRICO

3.1. ncclient

Es una librería de Python que facilita el desarrollo de aplicaciones y scripts por parte del cliente en torno al protocolo NETCONF. Fue desarrollo por Shikar Bhushan aunque ahora lo mantiene Leonidas Poulopoulos y Einar Nilsen-Nygaard [29].

3.1.1. Características

- Soporta todas las operaciones y capacidades definidas en el RFC 6241 [30].
- Permite solicitudes RPC [30].
- Mantienen a XML fuera del camino a menos que sea realmente necesario [30].
- Es extensible, es decir que se puede agregar fácilmente nuevas asignaciones de transporte, capacidades u operaciones [30].
- Tiene soporte para varios sistemas operativos que soportan NETCONF [31].

3.1.2. Requerimientos [29]

- Python 2.7 o Python 3.4+
- setuptools 0.6+
- Paramiko 1.7+
- lxml 3.3.0+
- libxml2
- libxslt

*Si se emplea ncclient en Debian/Ubuntu se necesita tener instaladas las librerías libxml2-dev y libxslt1-dev.

3.1.3. Instalación [29]

Puede emplear el siguiente comando

```
[ncclient] $ sudo python setup.py install
```

o vía pip

```
pip install ncclient
```

3.1.4. Manager

El objeto más común que se emplea con ncclient es la clase Manager. Una instancia de Manager es creada usando una función factory como connect_ssh o connect [30], y es aquella que permite enviar y recibir RPC de NETCONF desde y hacia a los dispositivos de red [31]. Un ejemplo genérico de la creación de instancias se puede apreciar en el Código 1.

```
1.  from ncclient import manager
2.  # Context manager keeping ncclient connection alive for the duration of
3.  # the context.
4.  with manager.connect(
5.      host='a.b.c.d',                 # IP address of device
6.      port=830,                       # Port to connect to
7.      username='user',                # SSH Username
8.      password='pass',                # SSH Password
9.      hostkey_verify=False            # Allow unknown hostkeys not in local store
10.     device_params={'name':'iosxe'}  # Device connection parameters
11. ) as m:                             # Context manager reference, i.e. instance of
    connected manager
12.     # Print out the NETCONF XML capabilities as reported by the device
13.     print(m.server_capabilities)
```

Código 1. Ejemplo genérico de la creación de una instancia Manager [31]

La clase Manager posee una serie de funciones integradas los cuales se muestran en la Tabla 1.

Tabla 1: Funciones de la clase Manager [30]

Métodos	Parámetros
get_config()	source, filter
edit_config()	target,config, default_operation, test_option y error_option
copy_config()	source,target
delete_config()	target
lock()/unlock()	target
get()	-
close_session()	-
kill_session()	sesión_id
commit()	confirmed, timeout, persist
discard_changes()	-
validate(source)	source
locked()	target
create_subscription()	filter,stream_name,start_time y stop_time
client_capabilities	-
server_capabilities	-
sesión_id	-
timeout	-
connected	-

4. TRABAJO PREPARATORIO

4.1. Revisar el marco teórico para la realización de la práctica.
4.2. Consultar que es un administrador de contexto en Python.
4.3. Describir los argumentos del método ncclient.transport.SSHSession.connect. **NOTA**: Tenga en cuenta que dicho método es utilizado por la función factory connect().
4.4. Consultar sobre el módulo xml.dom.minidom.

5. EQUIPO Y MATERIALES

- PC (GNS3)

6. PROCEDIMIENTO

6.1. Topología de red

a. Implementar la topología de red mostrada en la Figura 1.
NOTA: Emplear el proyecto de GNS3 creado en la práctica 2.

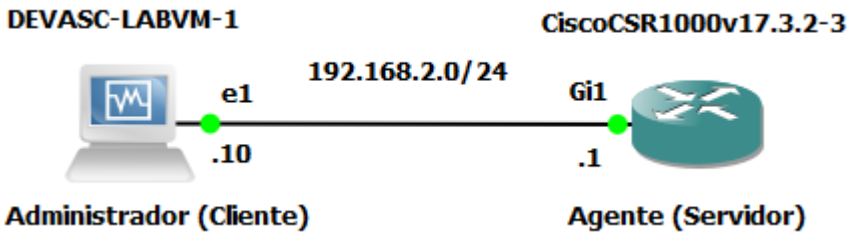


Figura 1. Topología de red

6.2. Inicio de sesión NETCONF en Agente con ncclient

a. Verificar si ncclient está instalado en la máquina DEVASC-LABVM

```
devasc@labvm:~$ pip3 list --format=columns | more
```

Obtendrá una salida como la siguiente:

```
Package                       Version
----------------------------- --------------
aiohttp                       3.6.2
ansible                       2.9.9
apache-libcloud               2.8.0
appdirs                       1.4.3
[SALIDA ELIMINADA]
ncclient                      0.6.7
netaddr                       0.7.19
netifaces                     0.10.4
[SALIDA ELIMINADA]
yarl                          1.4.2
zipp                          1.0.0
```

NOTA: En el caso de no encontrar ncclient en la lista, emplear el comando `pip3 install ncclient`.

b. Actualizar nccliente en la máquina DEVASC-LABVM.

```
devasc@labvm:~$ pip3 install --upgrade ncclient
```

c. Ingresar al directorio `labs/devnet-src/` y crear el subdirectorio `netconf`.

```
devasc@labvm:~$ cd labs/devnet-src/
devasc@labvm:~/labs/devnet-src$ mkdir netconf
```

d. Abrir Visual Studio Code (VSC), dirigirse al panel “Extensiones” e instalar la extensión “Python v2022.4.1”. (Ver Figura 2)

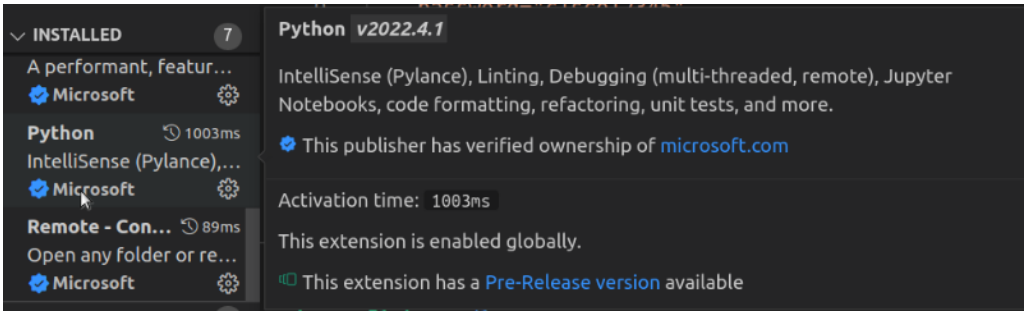


Figura 2. Extensión Python v2022.4.1

e. En panel “Explorar” dirigirse a la carpeta “DEVNET-SRC”, dar clic derecho en la subcarpeta “NETCONF” y crear el archivo *ncclient-connection-netconf.py.*

f. En el script importar la clase “manager” de la librería “ncclient”. Luego crear el objeto de sesión NETCONF “m” (Instancia de la clase “manager”). Este objeto actúa como un administrador de contexto y se crea al utilizar el método de conexión “connect()”. Dicho método incluye toda la información que es requerida para conectarse al servicio NETCONF que se ejecuta en el Agente. (ver Código 2)

NOTA: La instancia Manager del Código 2 muestra una forma poco común de usar un administrador de contexto.

```
1.  ##Programa para conexión al servicio NETCONF con SSH##
2.
3.  from ncclient import manager #Importar clase "manager"
4.
5.  ####Conexión SSH con constraseña(password)#####
6.
7.  #Administrador de contexto que mantiene "viva" la conexión
8.  # ncclient durante la duración del contexto.
9.  m=manager.connect( #Instancia de la clase “manager”(Objeto de sesión
    NETCONF)
10.     host="192.168.2.1", # Nombre de host o la dirección IP del dispositivo
11.     port="830", # Puerto para la conexión SSH con NETCONF
12.     username="admin1", # Nombre del cliente SSH para la autenticación
13.     #de usuario
14.     password="c1sco12345", # Contraseña del cliente SSH
15.     allow_agent=False, # No permite consular al agente SSH las claves
16.     look_for_keys=False, # No permite buscar en las ubicaciones habituales
17.     #(~/.shh) las claves del cliente
18.     hostkey_verify=False, # No verifica las claves de host contenidas
```

```
    #en ~/.ssh/known_hosts
    device_params={"name":"csr"} # Parámetros de los controladores
    #de dispositivos
)
'''
####Conexión SSH con clave pública####

#Administrador de contexto que mantiene "viva" la conexión
# ncclient durante la duración del contexto.

m=manager.connect(#Instancia de la clase “manager”(Objeto de sesión NETCONF
    host="192.168.2.1", # Nombre de host o la dirección IP del dispositivo
    port="830", # Puerto para la conexión SSH con NETCONF
    username="admin2", # Nombre del cliente SSH para la autenticación
    #de usuario
    hostkey_verify=False, # No verifica las claves de host contenidas
    # en ~/.ssh/known_hosts
    device_params={"name":"csr"} # Parámetros de los controladores
    # de dispositivos
)
'''
try:
    print('Sesión NETCONF: ', m.session_id) #Muestra sesión NETCONF

    '''

        # Aqui va el código para realizar
        # solicitudes al Agente.

    '''
finally:
    m.close_session() #Cierre de objeto de sesión NETCONF
```

Código 2. Conexión al servicio NETCONF con SSH [32]

Note que allow_agent y look_for_keys tienen el valor de False con el propósito de deshabilitar por completo la autenticación de clave pública. Es recomendable que hostkey_verify posea el valor True para tener seguridad de que el cliente se conecta al servidor SSH correcto. Debido a que en este laboratorio se tiene ambiente seguro se escribe el valor False para este parámetro. Por último, note que device_params contiene los controladores de los dispositivos. Su valor depende del equipo que se esté utilizando para el servicio NETCONF.

g. Guardar y probar el programa. Recuerde que se pude usar el botón “Run” ubicado en la parte superior derecha de VSC.

h. Comprobar que el Agente aceptó el requerimiento para una sesión NETCONF, para ello visualizar el mensaje %DMI-5-AUTH_PASSED generado en la consola del dispositivo. Recuerde que debe comprobar tanto la conexión SSH con contraseña como la de clave pública. Para ello deberá comentar parte del código en *ncclient-connection-netconf.py*

NOTA: Al emplear ncclient el intercambio de capacidades se realiza de forma automática. Es decir que se establece una sesión NETCONF de forma inmediata.

i. Crear un nuevo archivo llamado *ncclient-server-capabilities.py* en la subcarpeta NETCONF.

j. En el script se debe implementar la conexión SSH con clave pública, y listar las capacidades enviadas por el Agente. Para lograr listar las capacidades es necesario emplear la lista *m.server_capabilities*, que almacena las capacidades enviadas por el Agente.

NOTA: En el Código 3 se emplea un lazo *for* y la función *print* para listar las capacidades del dispositivo.

```
1.  ##Programa para conexión al servicio NETCONF con SSH y listar capacidades##
2.
3.  from ncclient import manager  # Importar clase "manager"
4.
5.  #Conexión SSH con clave pública
6.
7.  #Administrador de contexto que mantiene "viva" la conexión
8.  # ncclient durante la duración del contexto.
9.  with manager.connect(
10.     host="192.168.2.1", # Nombre de host o la dirección IP del dispositivo
11.     port="830", # Puerto para la conexión SSH con NETCONF
12.     username="admin2", # Nombre del cliente SSH para la autenticación
13.     #de usuario
14.     hostkey_verify=False, # No verifica las claves de host contenidas
15.     # en ~/.ssh/known_hosts
16.     device_params={"name":"csr"} # Parámetros de los controladores
17.     # de dispositivos
18. ) as m:#Instancia de la clase “manager” (Objeto de sesión NETCONF)
19.
20.     #Listar capacidades del agente (CSR1K)
21.     print("Lista de capacidades soportadas en CSR1K:\n")
22.     #Lazo for para para listar las capacidades almacenadas en
23.     #la lista m.server_capabilities
24.     for capability in m.server_capabilities:
25.         print(capability)
```

Código 3. Listar capacidades de Agente [32]

k. Guardar y probar el programa. Comente el resultado.

6.3. Recuperación de información con ncclient

En esta parte del laboratorio y en las siguientes se hará uso del módulo xml.dom.minidom para dar formato elegante a la salida XML.

a. Crear un nuevo archivo llamado *ncclient-get-config-netconf.py* en la subcarpeta NETCONF.

b. Emplear el método get_config() del objeto de sesión NETCONF “m”, para obtener la información de configuración en ejecución del Agente. Dicho método requiere el parámetro “*source*”, que es un *string* que especifica el almacén de datos NETCONF.

NOTA: En el Código 4 se comentaron las líneas que muestran en pantalla la lista de capacidades del Agente.

```
1.  ############################################################
2.  ###Programa para obtener la información de configuración
3.  ###del almacén de datos <running>
4.  ############################################################
5.
6.  from ncclient import manager  # Importar clase "manager"
7.
8.  ##Conexión SSH con clave pública
9.
10. #Administrador de contexto que mantiene "viva" la conexión
11. #ncclient durante la duración del contexto.
12. with manager.connect(
13.     host="192.168.2.1", # Nombre de host o la dirección IP del dispositivo
14.     port="830", # Puerto para la conexión SSH con NETCONF
15.     username="admin2", # Nombre del cliente SSH para la autenticación
16.     #de usuario
17.     hostkey_verify=False, # No verifica las claves de host contenidas
18.     # en ~/.ssh/known_hosts
19.     device_params={"name":"csr"} # Parámetros de los controladores
20.     # de dispositivos
21. ) as m:#Instancia de la clase “manager” (Objeto de sesión NETCONF)
22.     '''
23.     ##Listar capacidades del agente (CSR1K)
24.
25.     print("Lista de capacidades soportadas en CSR1K:\n")
26.     #Lazo for para para listar las capacidades almacenadas en
27.     #la lista m.server_capabilities
28.     for capability in m.server_capabilities:
29.         print(capability)
30.     '''
31.     ##Metódo get_config()
32.
33.     #Operación <get config> en almacén de datos <running>
34.     netconf_reply = m.get_config(source="running")
35.     #La variable netconf_reply almacena la respuesta del
36.     #servidor al finalizar la operación get_config()
37.
38.     print(netconf_reply) #Imprimir la variable netconf_reply
```

Código 4. Recuperar información de configuración de <running> [32]

c. Guardar y probar el programa. Comente el resultado. Puede usar XML Pretty Print Online Tool para visualizar mejor los datos recuperados.

d. Modificar el programa *ncclient-get-config-netconf.py.* Importe el módulo *xml.dom.minidom* y remplace *print(netconf_reply)* con la siguiente línea de código.

```
39. #Imprimir la variable netconf_reply con un formato XML más ordenado.
40. print(xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml())
```

Código 5. Imprimir respuesta del Agente en formato XML [32]

e. Guardar y probar el programa. Comente el resultado.

f. Crear un nuevo archivo llamado *ncclient-get-config-int-netconf.py* en la subcarpeta NETCONF.

NOTA: Para no empezar el nuevo script desde cero, puede copiar todas las líneas de código de *ncclient-get-config-netconf.py*

g. Crear una variable llamada *netconf_filter* que contenga el elemento <filter>. Recuerde que <filter> permite extraer todo o parte de un modelo de datos específico contenido en un almacén de datos. En este caso usar <filter> para extraer la información de configuración de la interfaz Gigabit Ethernet 1 contenida en el modelo de datos ietf-interfaces. (ver Código 6)

```
35.     #Elemento <filter> para extraer la información de configuración de la
36.     #interface GigabitEthernet1 contenida en el modelo de datos
37.     #ietf-interfaces.
38.     netconf_filter = """
39.     <filter xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
40.       <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
41.         <interface>
42.           <name>GigabitEthernet1</name>
43.         </interface>
44.       </interfaces>
45.     </filter>
46.       """
```

Código 6. Variable con elemento <filter>

h. Modificar el método get_config(). Para ello añadir el parámetro “filter” y asociarle la variable *netconf_filter.* (ver Código 7)

```
47.     #Operación <get config> en almácen de datos <running> y usando
48.     #elemento <filter>
49.     netconf_reply = m.get_config(source="running",filter= netconf_filter)
```

Código 7. Método get_config() [32]

i. Guardar y probar el programa. Comente el resultado.

NOTA: En el Código 6, el elemento <filter> contiene el namespace de NETCONF: *urn:ietf:params:xml:ns:netconf:base:1.0*. Esto debido a que el parámetro filter del método get_config() no asume dicho namespace y en el caso de no especificar el mismo se producirá un error. Esta falla se presenta en los dispositivos IOS XE 17.x o superior.

Una alternativa adicional para no caer en dicho error es eliminar el elemento <filter> en la variable *netconf_filter* y especificar el tipo de filtro “subtree” en el parámetro filter del método get_config(), tal como se muestra en el Código 8.

```
#Filtro “subtree” para extraer la información de configuración de la
#interface GigabitEthernet1 contenida en el modelo de datos
#ietf-interfaces.
netconf_filter = """
      <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
        <interface>
          <name>GigabitEthernet1</name>
        </interface>
      </interfaces>
      """

#Operación <get config> en almácen de datos <running> y
#usando un filtro del tipo “subtree”
netconf_reply =
    m.get_config(source="running",filter=('subtree',netconf_filter))
```

Código 8. Método get_config() con filtro “subtree”

j. Importar el módulo *xmltodict* y usar la siguiente línea de código (ver Código 9) para convertir la salida XML en un diccionario ordenado de Python.

NOTA: Un diccionario ordenado puede ser fácilmente manipulado en la programación con Python.

```
# Convertir datos XML en Diccionario Ordenado de Python
netconf_data= xmltodict.parse(netconf_reply.xml)["rpc-reply"]["data"]
```

Código 9. XML a diccionario ordenado de Python [4]

k. Listar la información de configuración de la interfaz Gigabit Ethernet 1. (Ver Código 10)

```
# Creamos un diccionario ordenado de las interfaces
interface=netconf_data["interfaces"]["interface"]
# Debido a que solo hay datos de una interfaz
# se convierte en una diccionario de la información
# de la interfaz

#Accedemos a los valores del diccionario.
print("Interfaz: ",interface["name"])
print("Estatus: ", interface["enabled"])
print("Tipo de interfaz: ", interface["type"]["#text"])
print("Dirección IPv4: ", interface["ipv4"]["address"]["ip"])
print("Mascara de red IPv4: ",
    interface["ipv4"]["address"]["netmask"])
```

Código 10. Listar información de interfaz

6.4. Configuración del Agente con ncclient

a. Crear un nuevo archivo llamado *ncclient-edit-config-netconf.py* en la subcarpeta NETCONF. Puede reutilizar los programas ya implementados.

b. Crear la variable *netconf_banner* que contenga el elemento <config>. Recuerde que <config> contiene la estructura de datos a configurar, la cual es específica de un modelo de datos. En este caso emplear <config> para configurar el mensaje del día del Agente empleando el modelo de datos CISCO-IOX-XE-native. (ver Código 11)

```
33.     ##Metódo edit_config()
34.
35.     #Elemento <config> para editar el banner del CSR1k
36.     #especificada en el modelo de datos
37.     #CISCO-IOX-XE-native.
38.     netconf_banner = """
39.     <config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
40.       <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
41.         <banner>
42.           <motd>
43.             <banner>Configuración NETCONF-PYTHON</banner>
44.           </motd>
45.         </banner>
46.       </native>
47.     </config>
48.     """
```

Código 11. Variable con elemento <config>

c. Emplear el método edit_config() del objeto de sesión NETCONF para configurar el mensaje del día en el almacén de datos de configuración <candidate>. (ver Código 12)

Tome en cuenta que edit_config() admite los siguientes parámetros:

- **target (*string*):** Almacén de datos de configuración a modificar.
- **config (*string*):** Variable con el elemento <config>.
- **default_operation (*string*):** La forma de funcionamiento de la operación <edit-config>. Puede tomar el valor de “none”, “marge” o “replace”.
- **test_option (*string*):** Las pruebas de validación a implementar. Puede tomar el valor de “none”, “test-then-set”, “set” o “test-only”.
- **error_option (*string*):** Las opciones de error a implementar. Puede tomar el valor de “none”, “stop-on-error”, “continue-on-error”, o “rollback-on-error”.

```
49.     #Operación <edit config> en almacén de datos <candidate> y usando
50.     #elemento <config>
51.     netconf_reply = m.edit_config(target="candidate",config= netconf_banner)
```

Código 12. Método edit_config() [32]

d. Guardar y probar el programa. Comente el resultado.

e. Generar un programa para crear una interfaz de loopback en el almacén de datos de configuración <candidate> empleando el método edit_config(). Recuerde probar el mismo.

NOTA: El modelo de datos *CISCO-IOX-XE-native* utiliza el submódulo *Cisco-IOS-XE-interfaces* para definir la estructura de datos de las interfaces del CSR1K donde están incluidas las de loopback. Recuerde que existen varios modelos de datos

como los IETF o OPENCONFIG. En esta ocasión se está empleando los modelos NATIVOS de CISCO XE.
Si ve conveniente puede descargar los modelos *CISCO-IOX-XE-native* y *Cisco-IOS-XE-interfaces* del repositorio de GITHUB, y emplear el comando: `pyang -f tree Cisco-IOS-XE-native.yang` para visualizar de mejor manera el nodo container “Interfaces” que incluye el nodo list “loopback”.

f. Una solución al problema del literal “e” se muestra en el Código 13. Analizar el mismo y compararlo con el código creado en el literal anterior.

```
1.  ############################################################
2.  ###Programa para crear una interfaz loopback en CSR1k en
3.  ###almacén de datos <candidate>
4.  ############################################################
5.
6.  from ncclient import manager  # Importar clase "manager"
7.  import xml.dom.minidom #Importar el módulo xml.dom.minidom
8.
9.
10. ##Conexión SSH con clave pública
11.
12. #Administrador de contexto que mantiene "viva" la conexión
13. #ncclient durante la duración del contexto.
14. with manager.connect(
15.     host="192.168.2.1", # Nombre de host o la dirección IP del dispositivo
16.     port="830", # Puerto para la conexión SSH con NETCONF
17.     username="admin2", # Nombre del cliente SSH para la autenticación
18.     #de usuario
19.     hostkey_verify=False, # No verifica las claves de host contenidas
20.     # en ~/.ssh/known_hosts
21.     device_params={"name":"csr"} # Parámetros de los controladores
22.     # de dispositivos
23. ) as m:#Instancia de la clase “manager” (Objeto de sesión NETCONF)
24.     '''
25.     ##Listar capacidades del agente (CSR1K)
26.
27.     print("Lista de capacidades soportadas en CSR1K:\n")
28.     #Lazo for para para listar las capacidades almacenadas en
29.     #la lista m.server_capabilities
30.     for capability in m.server_capabilities:
31.         print(capability)
32.     '''
33.     ##Metódo edit_config()
34.
35.     #Elemento <config> para crear una interfaz loopback en CSR1k
36.     #especificada en el modelo de datos
37.     #CISCO-IOX-XE-native.
38.     config_loopback = """
39.     <config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
40.       <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
41.         <interface>
42.           <Loopback>
43.             <name>{nombre}</name>
44.             <description>{descripcion}</description>
45.             <ip>
46.               <address>
47.                 <primary>
48.                   <address>{direccion_IP}</address>
49.                   <mask>{mascara_red}</mask>
50.                 </primary>
51.               </address>
52.             </ip>
```

```
          </Loopback>
        </interface>
      </native>
    </config>
    """


    print("Configuración de Loopback")
    #Se pide al usuario la información de la nueva interfaz Loopback
    #La información se almacena en el diccionario new_loopback
    new_loopback={}
    new_loopback["name"]=input("Número de la interfaz Loopback a crear: ")
    new_loopback["desc"]=input("Descripción de la interfaz Loopback: ")
    new_loopback["ip_address"]=input("Dirección IP: ")
    new_loopback["mask"]=input("Mascara de red: ")

    #Se asignan los elementos del diccionario al elemento <config>
    data_loopback=config_loopback.format(
        nombre=new_loopback["name"],
        descripcion=new_loopback["desc"],
        direccion_IP=new_loopback["ip_address"],
        mascara_red=new_loopback["mask"]
    )

    #Operación <edit config> en almacén de datos <candidate> y usando
    #elemento <config>
    netconf_reply = m.edit_config(target="candidate",config=data_loopback)
    #La variable netconf_reply almacena la respuesta del
    #servidor al finalizar la operación edit_config()

    #Imprimir la variable netconf_reply con un formato XML más ordenado.
    print(xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml())
```

Código 13. Crear una interfaz loopback en <candidate> [4]

g. Guardar y probar el programa del Código 13. Comente los resultados obtenidos.

6.5. Eliminación de información de configuración con ncclient

Para poder eliminar parte de la configuración de un almacén de datos se emplea el método edit_config() con la operación <delete>, pero en el caso de que se requiera eliminar toda la configuración de un almacén de datos se emplea el método delete_config ().

a. Crear un archivo llamado *ncclient-delete-loopback-netconf.py* en la subcarpeta de NETCONF. Puede reutilizar los programas ya implementados.

b. Utilizar edit_config() para eliminar la interfaz de loopback creada en la sección anterior. Una manera de realizarlo se muestra en el Código 14.

```
    ##Metódo edit_config()

    #Elemento <config> con operación "delete" para eliminar una
    #interfaz loopback en CSR1k especificada en el modelo de datos
    #CISCO-IOX-XE-native.
    config_loopback = """
    <config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
        <interface>
          <Loopback operation="delete">
            <name>{nombre}</name>
          </Loopback>
```

```
        </interface>
      </native>
    </config>
    """


    print("Configuración de Loopback")
    #Se pide al usuario el número de la interfaz Loopback a eliminar
    #La información se almacena en el diccionario new_loopback
    new_loopback={}
    new_loopback["name"]=input("Número de la interfaz Loopback a eliminar: ")


    #Se asigan el elemento del diccionario al elemento <config>
    data_loopback=config_loopback.format(
        nombre=new_loopback["name"]
    )

    #Operación <edit config> en almacén de datos <candidate> y usando
    #elemento <config>
    netconf_reply = m.edit_config(target="candidate",config=data_loopback)
    #La variable netconf_reply almacena la respuesta del
    #servidor al finalizar la operación edit_config()

    #Imprimir la variable netconf_reply con un formato XML más ordenado.
    print(xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml
```

Código 14. Eliminar una interfaz loopback de <candidate> [4]

c. Guardar y probar el programa *ncclient-delete-loopback-netconf.py*. Comente el resultado.

d. Crear un nuevo archivo llamado *ncclient-delete-config-netconf.py* en la subcarpeta NETCONF. Puede reutilizar los programas ya implementados.

e. Emplear el método delete_config() del objeto de sesión NETCONF para eliminar la configuración del almacén de datos <running>. Dicho método requiere el parámetro “target”, que es un *string* que especifica un almacén de datos de configuración de NETCONF. (ver Código 15)

```
##Metódo delete_config()

#Operación <delete config> en almacén de datos <runnnig>
netconf_reply = m.delete_config(target="running")
#La variable netconf_reply almacena la respuesta del
#servidor al finalizar la operación delete_config()

#Imprimir la variable netconf_reply con un formato XML más ordenado
print(xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml()
```

Código 15. Método delete_config

f. Guardar y probar el programa creado. Comente el resultado.

6.6. Guardar configuración del Agente con ncclient

Una de las ventajas de NETCONF es que permite a los proveedores crear operaciones RPC nativas para ciertas actividades que son específicas de su

plataforma. Por ejemplo, los dispositivos IOS XE pueden hacer uso de la operación <save-config> ya que es parte de uno de los modelos de datos nativos. Dicha operación permite guardar la configuración en ejecución (running configuration) en la configuración de inicio (startup configuration).

a. Crear un nuevo archivo llamado *ncclient-save-config-netconf.py* en la subcarpeta NETCONF. Puede reutilizar los programas ya implementados.

b. En el script importar la clase “xml_” de la librería “ncclient”. Esta clase permitirá crear un RPC personalizado. (ver Código 16)

```
8. from ncclient import xml_ # Importar clase "xml_"
```

Código 16. Importar clase xml_

c. Emplear el método commit() del objeto de sesión NETCONF para que el dispositivo emplee los datos de configuración contenidos en el almacén de datos <candidate> como la configuración en ejecución. No es necesario confirmar commit. (ver Código 17)

```
34. #Operación <commit> sin emplear confirmación
35. netconf_reply = m.commit(confirmed=False)
36. #La variable netconf_reply almacena la respuesta del
37. #servidor al finalizar la operación commit()
```

Código 17. Método commit()

NOTA: Este paso se puede omitir si el almacén de datos <candidate> no está habilitado ya que se puede realizar la configuración directamente sobre el almacén de datos <running>.

d. Crear la variable *netconf_save* que contenga el RPC <save-config> del modelo de datos cisco-ia. Note que en esta ocasión en lugar de crear variables con elementos <filter> o <config>, llamamos explícitamente al RPC desde un modelo de datos. (Ver Código 18)

```
38. #Llamamos explícitamente al RPC <save-config> desde
39. #el modelo de datos Cisco-ia
40. netconf_save="""
41. <save-config xmlns="http://cisco.com/yang/cisco-ia"/>
42. """
```

Código 18. Variable que contiene <save-config> [4]

e. Emplear el método dispatch() del objeto de sesión NETCONF para enviar la operación personalizada. (Ver Código 19)

```
#Se usa el metódo dispatch() para enviar al servidor
#el RPC personalizado <save-config>
netconf_reply_d = m.dispatch(xml_.to_ele(netconf_save))
#La variable netconf_reply_d almacena la respuesta del
#servidor al finalizar la operación dispatch()

#Imprimir la variable netconf_reply_d con un formato XML más ordenado
print(xml.dom.minidom.parseString(netconf_reply_d.xml).toprettyxml())
```

Código 19. Método dispatch() [4]

f. Guardar y probar el programa. Comente el resultado.

NOTA: Debe apagar y encender de nuevo el router. Luego, mediante la CLI deberá observar que el mensaje del día configurado en la sección 6.4 este presente al iniciar el dispositivo. Explique por qué sucedió lo mencionado anteriormente.

7. INFORME

7.1. Presentar capturas de pantalla del procedimiento realizado en la práctica con la debida explicación de ser necesario.
NOTA: Algunos enunciados en el procedimiento, requieren respuesta o solicitan algún análisis. Recuerde incluir los mismos.

7.2. Configure OSPF usando ncclient. Tenga en cuenta la topología de red de la Figura 3.
NOTA: Debe emplear el modelo de datos Cisco-IOS-XE-ospf, específicamente enfocarse en la declaración augment "/ios:native/ios:router"

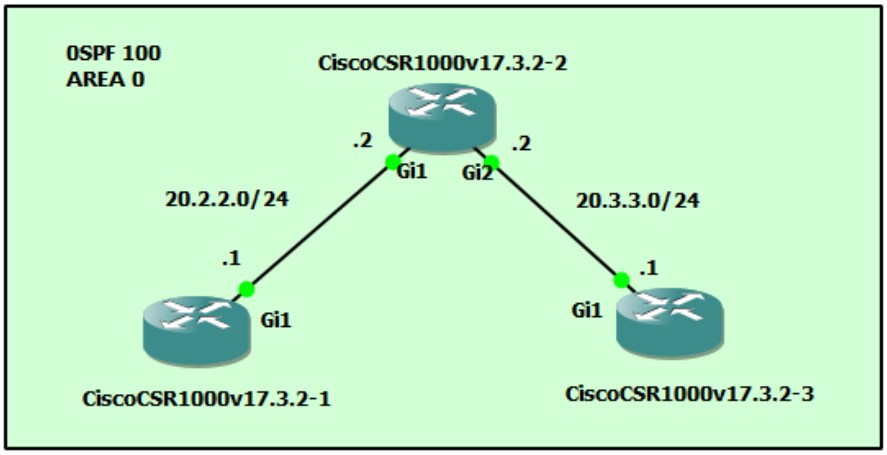


Figura 3. Topología de red para OSPF

7.3. Conclusiones y Recomendaciones.

8. REFERENCIAS

Las referencias de la práctica 5 se encuentran en el apartado de referencias de este documento.

ANEXO VI

Link que contiene los reactivos en formato Moodle:

https://epnecuador-my.sharepoint.com/:f:/g/personal/alex_guaman01_epn_edu_ec/ErMuM7GHHYxCjR6r-hA6R-sBIp4JkJ5gqPg7dF2bV4rMOQ

Link que contiene los videos de los procedimientos de las prácticas:

https://epnecuador-my.sharepoint.com/:f:/g/personal/alex_guaman01_epn_edu_ec/EoCaiO__EWBLkIVF2sbCfUYBULn0ynyuoqk1zSSV3HYs7w?e=dCjSPS

ANEXO VII

PREPARATORIO DE LA PRÁCTICA 1

NOTA: Los literales del 1 al 5 no se presentan en esta resolución. Los estudiantes deberán descargar cada uno los archivos/softwares solicitados en los mismos.

6. Consultar los requisitos mínimos de vCPU, tamaño de memoria RAM y disco duro virtual para la instalación de la imagen de disco del router CISCO CSR1000v versión IOS XE 17.3.2 en entornos KVM.

   RESPUESTA: Para instalar Cisco IOS XE 17.3.2 en entornos Linux/KVM (Kernel-based Virtual Machine), se necesita de una máquina virtual y la instalación mediante archivos del tipo .iso o .qcow2. Los requerimientos mínimos son 1vCPU con al menos 4GB RAM y 8 GB de tamaño de disco duro virtual [14].

7. Consultar y describir 3 dispositivos (routers o switches) que soporten el protocolo NETCONF.
   La siguiente respuesta no es genérica, puede cambiar de acuerdo a lo escogido por el estudiante. Algunos dispositivos se pueden encontrar en el ANEXO IV.

   RESPUESTA:

   - ***Router NetEngine AR6710-L26T2X4*** [33]
     Router empresarial Huawei de última generación perteneciente a la serie NetEngine AR6700. Este modelo, así como los otros que conforman la serie ofrecen un rendimiento de reenvío más alto que el promedio de la industria ya que emplean procesadores multinúcleo de alto rendimiento. Son usados comúnmente para soluciones SD-WAN.

Figura 1. NetEngine AR6710-L26T2X4

Las características más importantes de dicho dispositivo se muestran a continuación:

- Puertos WAN: 2 x GE eléctricos, 2 x 10GE ópticos
- Puertos LAN: 24 x GE eléctricos
- Dimensiones (H x W x D): 44.4 mm x 442.0 mm x 420.0 mm

- Procesador: ARM64 (4-core)
- Memoria: 4GB
- Flash: 2GB
- Administración y Mantenimiento: GTP, SNMP(v1/v2/v3), RMON, NTP, CLI, NETCONF/YANG, TCP FPM, NQA, entre otros.
- Seguridad: ACL, Autenticación AAA, Autenticación RADIUS, PKI, seguridad ARP, entre otros.

Una mayor especificación de las características del dispositivo se puede encontrar en [33].

- ***Switch Catalyst 3650*** [34]

  Switch de acceso Cisco de la clase empresarial que provee la base para la convergencia total entre la parte cableada e inalámbrica en una única plataforma.

Figura 2. Catalyst 3650

Las características más importantes de dicho dispositivo se muestran a continuación:

- Puertos Ethernet: 24 o 48 a 10/100/1000 Mbps PoE+
- DRAM: 4GB
- Flash: 2GB o 4GB
- Software: Cisco IOS XE
- Seguridad: ACL, Autenticación TACACS+ y RADIUS, BPDU, STRG, entre otros.
- Administración y Mantenimiento: NETCONF, CLI, SNMP(v1/v2/v3), entre otros.

Una mayor especificación de las características del dispositivo se puede encontrar en [34].

- ***Router ACX7100-32C*** [35]

  Router Juniper perteneciente a la serie ACX7100 completamente equipado para uso metropolitano, de grandes empresas y de centros de datos. La serie ACX7100 ofrece beneficios en cuanto a capacidad, densidad y baja latencia.

Figura 3. ACX7100-32C

Las características más importantes de dicho dispositivo se muestran a continuación:

- Puertos: 32 de 100GbE y 4 de 400GbE QSFP56-DD.
- Dimensiones (H x W x D): 44.09 x 4.45 x 59.49 cm.
- Seguridad: MACsec, SSH, Secure boot, AAA (Authentication, authorization, and accounting).
- Administración y Automatización: ZTP, NETCONF-YANG, Openconfig, Python scripts.

Una mayor especificación de las características del dispositivo se puede encontrar en [35].

8. Consultar sobre la estructura de red básica de NETCONF.

RESPUESTA:

Los elementos de la arquitectura de red básica de NETCONF son los siguientes:

- ***Cliente (Administrador):*** Es el encargado de administrar el dispositivo de red. Por ejemplo, envía solicitudes <rpc> para consultar o modificar uno o varios valores de parámetros contenidos en el dispositivo. Así también es capaz de aprender el estado del dispositivo gracias a la configuración de alarmas o eventos [36].
- ***Servidor (Agente):*** Es el dispositivo de red que responde a las solicitudes iniciadas por el cliente. El Agente al recibir una solicitud de un cliente NETCONF, analiza dicha solicitud y envía una respuesta al cliente [36].

PREPARATORIO DE LA PRÁCTICA 2

NOTA: El literal 1 no contiene una resolución. Dependerá del estudiante revisar el marco teórico expuesto en la práctica.

2. Consultar y responder a las siguientes preguntas.

- ¿NETCONF podría superponerse sobre otro protocolo de transporte a parte de SSH?

RESPUESTA:

Si, se pude superponer sobre cualquier protocolo que proporcioné un mecanismo para indicar el tipo de sesión (cliente o servidor) a la capa de transporte seguro de NETCONF [7]. En el RFC 6241, no se define un protocolo de transporte específico a utilizar, sin embargo, se detalla ciertos requisitos que son mostrados a continuación.

1. ***Orientado a la conexión*** [7]

   El protocolo de transporte debe ofrecer una operación orientada a la conexión, por las siguientes razones:

   - NETCONF necesita una comunicación permanente entre cliente y servidor.
   - La conexión NETCONF debe entregar datos de forma confiable y secuencial.
   - La recuperación de fallos debe ser simple y robusta.

2. ***Autenticación, integridad y confidencialidad*** [7]

   El protocolo de transporte debe permitir que las conexiones NETCONF ofrezcan autenticación, integridad de datos, confidencialidad y protección de repetición (Replay Protection). Tanto cliente como servidor NETCONF deben regirse a los mecanismos de seguridad, confidencialidad y autenticación establecidos por el protocolo de transporte escogido.

- ¿Quién es responsable de la autenticación de usuarios NETCONF?

RESPUESTA:

El método de autenticación empleado por el protocolo de transporte es el encargado de realizar la autenticación de usuarios NETCONF. Esto quiere decir que el protocolo de transporte, independientemente del método de autenticación que escoja, será el encargado de realizar la autenticación del servidor al cliente y del cliente al servidor. Lo único que realiza NETCONF, es asumir que dicho proceso de autenticación es lo suficientemente confiable y lo emplea para obtener una identidad del cliente autenticado que se denomina usuario NETCONF [7].

- ¿Qué es AAA (Authentication, Authorization, Accounting)?

  RESPUESTA:
  AAA es un conjunto de servicios que permiten controlar el acceso a los recursos informáticos, establecer políticas, evaluar el uso y proporcionar la información necesaria para contabilizar los servicios. Dichos procesos se consideran importantes para una gestión y seguridad de la red eficaz [37].
  Autenticación (Authentication): Proporciona una manera de identificar a un usuario. Por ejemplo, haciendo al usuario ingresar un nombre y contraseña antes de otorgar el acceso a un recurso [37].
  Autorización (Authorization): Hace cumplir las políticas, es decir determina qué tipos de actividades, recursos o servicios le son permitidos acceder a un usuario [37].
  Contabilidad (Accounting): Permite medir y registrar los recursos que consume un usuario durante el acceso. Por ejemplo, permite contabilizar la cantidad de tiempo del sistema o la cantidad de datos que un usuario ha enviado o recibido durante una sesión [37].

- ¿Qué es un servidor TACACS+?

  RESPUESTA:
  Un servidor TACACS+ es un servidor externo que se utiliza para validar a los usuarios el acceso a un dispositivo de red o servidor de acceso a la red. Un servidor TACACS+ permite que un único servidor de control de acceso (puto centralizado) denominado demonio TACACS+ proporcione el servicio de autenticación, autorización y contabilidad de forma independiente a todos los usuarios que intentan obtener acceso a un dispositivo de red o a un servicio de red [38].

- ¿Qué es ssh-keygen? y ¿Cómo se emplea para la autenticación de usuario con SSH?

  RESPUESTA:
  ssh-keygen es una herramienta para crear nuevos pares de claves (pública y privada) de autenticación para SSH. Dichas claves suelen ser usadas para autenticar hosts o usuarios [39].
  El protocolo SSH emplea las claves creadas con ssh-keygen en el método de autenticación de usuario de clave pública. Este método se considera más seguro

debido a que evita la necesidad de almacenar contraseñas en archivos, así también elimina la posibilidad de que un servidor comprometido robe las contraseñas de los usuarios [39].

PREPARATORIO DE LA PRÁCTICA 3

NOTA: El literal 1 no contiene una resolución. Dependerá del estudiante revisar el marco teórico expuesto en la práctica.

2. Responda a los siguientes enunciados:

- Describa los modelos de datos YANG de IETF, OPENCONFIG y nativos.

RESPUESTA:

Los modelos de datos provienen de dos grupos: los organismos de estándares o grupos comunitarios como IETF y OPENCONFIG; y los proveedores de software y equipos de red como lo son Cisco, Juniper, Huawei, entre otros [40].

- ***Modelos IETF***: El organismo de estándares IETF a parte de generar otros estándares como YANG, NETCONF y RESCONF, también se encarga de la creación de modelos de datos a los que se les denomina modelos IETF [40].
- ***Modelos OPENCONFIG***: Son los modelos creados por OPENCONFIG, que es un grupo de trabajo informal de operadores de red. La función inicial de dicho grupo es desarrollar modelos de datos YANG independientes del proveedor que cumplan las necesidades y requisitos de múltiples operadores de red [41].
- ***Modelos nativos***: Son los modelos de datos YANG creados por los proveedores. Se denominan modelos de datos “Nativos”, debido a que son originarios de los dispositivos/softwares con los que se encuentran asociados [40].

- ¿Qué es un espacio de nombres XML (*XML namespace*) y qué relación tiene con un módulo YANG?

RESPUESTA:

Un XML namespace es aquel que contine sus propios nombres de elementos y atributos. Dicho namespace es único y se identifica mediante una referencia URI (Uniform Resource Identifiers) [11].

Un módulo YANG está vinculado a un XML namespace particular, que es identificado por un URI global único. Tanto cliente como servidor NETCONF emplean el namespace durante la codificación XML de los datos [12].

Un XML namespace es asignado a un módulo por la organización propietaria del mismo. Es importante recalcar que el URI de dicho namespace debe ser elegido de tal forma que no entre en conflicto con otro de otra empresa u organización [12].

En el lenguaje YANG se emplea la declaración `namespace` para definir el XML namespace que identifica al módulo YANG. [12].

- ¿Qué es y para que se emplea declaración YANG: `prefix`?

RESPUESTA:

La declaración `prefix` es aquella que permite definir un valor (string) que sirva de prefijo para acceder a un módulo. Dicho prefijo es único y está asociado a un módulo y namespace especifico. Se puede usar para referirse a los nodos contenidos en un módulo. Por ejemplo "`if:ifName`", indica que el nodo `ifName` está asociado al módulo que tiene el prefijo `if` [12].

- ¿Qué es pyang y para qué es empleado?

RESPUESTA:

Pyang es un validador, transformador y generador de código YANG, escrito en lenguaje de programación Python. Se emplea para validar la corrección de los módulos YANG, para transformar módulos YANG en distintos formatos y también para escribir complementos para crear código a partir de los módulos [42].

PREPARATORIO DE LA PRÁCTICA 4

NOTA: El literal 1 no contiene una resolución. Dependerá del estudiante revisar el marco teórico expuesto en la práctica.

2. Consultar los valores del atributo “operation” disponibles en la operación base <edit-config>.

   RESPUESTA:

   El atributo “operation” posee los siguientes valores[7]:

   - ***marge***: Es el comportamiento predeterminado de la operación <edit-config>. Los datos que serán configurados se fusionan en el nivel correspondiente con la configuración establecida en el almacén de datos de configuración identificado por el elemento <source>.
   - ***replace***: Los datos de configuración que se identifican por el elemento que contiene dicho atributo se reemplazan por cualquier configuración establecida en un almacén de datos. Si no existen tales datos de configuración en el almacén estos se crean.
   - ***create***: Los datos de configuración que se identifican por el elemento que contiene dicho atributo se agregan a la configuración solo si los datos de configuración aún no existen en el almacén de datos, caso contrario se producirá un error.
   - ***delete***: Los datos de configuración que se identifican por el elemento que contiene dicho atributo se eliminan de configuración solo si los datos de configuración existen en el almacén de datos, caso contrario se producirá un error.
   - ***remove***: Los datos de configuración que se identifican por el elemento que contiene dicho atributo se eliminan de configuración solo si los datos de configuración existen en el almacén de datos, caso contrario el servidor ignora silenciosamente la operación “remove”.

3. Consultar los valores de que pueden tomar <test-option> y <error-option> en la operación base <edit-config>.

   RESPUESTA:

   Los valores mostrados a continuación, solo se pueden asociar al elemento <test-option> en el caso de que el dispositivo haya anuncio la capacidad Validate:1:1 [7].

- ***<test-then-set>:*** Es el valor predeterminado. Realiza una prueba de validación antes de intentar establecer la configuración solicitada. Si existen errores no ejecuta la operación <edit-config>.
- ***<set>:*** Establece la configuración sin una prueba de validación.
- ***<test-only>:*** Realiza una prueba de validación, sin intentar configurar.

El elemento <option-error> puede tomar los siguientes valores [7]:

- ***<stop-on-error>:*** Es el valor predeterminado. Al primer error anula la operación <edit-config>.
- ***<continue-on-error>:*** En caso de producirse un error, lo registrar y genera una respuesta sin embargo continúa procesando los datos de configuración.
- ***<rollback-on-error>:*** En el caso de producirse un error, el servidor detendrá el procesamiento de la operación <edit-config> y restaurará la configuración especificada a su estado inicial. Este valor solo está disponible si el dispositivo anuncio la capacidad Rollback-on-Error.

4. Consultar qué información se muestra en el elemento <rpc-error>.

RESPUESTA:

Un elemento <rpc-error> puede contener la siguiente información [7]:

- ***error-type:*** Define la capa donde se ha producido el error.
  - ***transport:*** Para la capa de transporte seguro.
  - ***rpc:*** Para la capa de mensajes.
  - ***protocol:*** Para la capa de operaciones.
  - ***application:*** Para la capa de contenido.
- ***error-tag:*** Contiene un *string* que permite identificar el tipo el error.
- ***error-severity:*** Contiene un *string* que muestra la severidad del error. Puede tomar el valor de ***error*** o ***warning***.
- ***error-app-tag:*** Muestra un *string* en el caso de producirse un error especifico de un modelo de datos o de una implementación,
- ***error-path:*** Contiene la ruta (xPath) que identifica el lugar (nodo-elemento) donde se produjo el error.
- ***error-message:*** Describe el error, de forma que puede ser entendido por los usuarios.
- ***error-info:*** Muestra el contendido de error especifico de protocolo o del modelo de datos.

5. Consultar sobre la operación <commit>.

RESPUESTA:

La operación <commit> permite establecer la configuración de almacén de datos <candidate> como la nueva configuración de ejecución del dispositivo. Un administrador puede realizar cualquier configuración en <candidate> sin afectar la configuración en ejecución del dispositivo. Una vez este seguro de haber terminado o completado la configuración en <candidate> podrá emplear <commit> para solicitar que el dispositivo empleé dicha configuración como la configuración en ejecución del dispositivo [7].

Si el dispositivo anuncia la capacidad Confirmed-commit, la operación <commit> admite nuevos parámetros, los cuales son mostrados a continuación [7]:

- <confirmed>: Establece que debe existir una confirmación para la operación <commit> cada cierto periodo de tiempo, caso contrario se revertirá la operación.
- <confirm-timeout>: Permite configurar el tiempo límite en segundos para poder realizar la confirmación de <commit>. En el caso de realizar la confirmación este tiempo se restablece a su valor configurado.
- <persist>: Permite asignar un token a la operación <commit> confirmada. Proporciona una confirmación de seguimiento y una confirmación de <commit> en cualquier sesión.
- <persist-id>: Corresponde al valor configurado en <persist>. Se emplea para realizar seguimiento a la operación <commit> confirmada.

PREPARATORIO DE LA PRÁCTICA 4

NOTA: El literal 1 no contiene una resolución. Dependerá del estudiante revisar el marco teórico expuesto en la práctica.

2. Consultar que es un administrador de contexto en Python.

RESPUESTA:

Un administrador de contexto permite asignar y liberar recursos cuando lo desee. El ejemplo más utilizado de un administrador de contexto se produce al emplear la declaración `with` (ver Código 1). Un administrador de contexto permite ejecutar como un par dos operaciones relacionadas, con un bloque de código en el medio. Note que el Código 1 abre un archivo, escribe datos en él y luego lo cierra. En el caso de producirse un error al escribir los datos en el archivo, este intentara cerrarlo. El Código 2 es equivalente al Código 1 y es la forma menos común de usar un administrador de contexto [43].

```
1.  with open('some_file', 'w') as opened_file:
2.      opened_file.write('Hola!')
```

Código 1. Administrador de contexto con declaración `with` [43]

```
1.  file = open('some_file', 'w')
2.  try:
3.      file.write('Hola!')
4.  finally:
5.      file.close()
```

Código 2. Administrador de contexto sin declaración `with` [43]

Si se compara ambos códigos, la mayor diferencia es que el Código 1 emplea una menor cantidad de líneas de código. La principal ventaja de usar la declaración `with` es que asegura que nuestro archivo se cerrará sin prestar atención a como se ejecuta el bloque anidado [43].

3. Describir los argumentos del método ncclient.transport.SSHSession.connect().
NOTA: Tenga en cuenta que dicho método es utilizado por la función factory connect().

RESPUESTA [30]**:**

- ***host:*** nombre de host o la dirección IP a conectarse.
- ***port:*** puerto para la conexión NETCONF.
- ***timeout:*** tiempo de espera para conexión de socket.

- ***unknown_host_cb:*** se emplea cuando no se reconoce la clave de host del servidor. Posee dos argumentos: hostname y figerprint.
- ***username:*** usuario para la autenticación SSH.
- ***password:*** contraseña que se utiliza si se ocupa el método de autenticación por contraseña, o la contraseña que se utiliza para desbloquear las claves que lo requieran.
- ***key-filename:*** nombre del archivo donde se puede encontrar la clave privada a utilizar.
- ***allow_agent:*** permite consultar las claves disponibles en el Agente.
- ***hostkey_verify:*** permite verificar las claves de host en ~/.ssh/known_hosts
- ***hostkey_b64:*** establece que la conexión será exitosa cuando la clave de hots pública del servidor coincida con estas claves: etc/ssh/ssh_host_*pub o ssh-keyscan.
- ***look_for_keys:*** permite buscar en las ubicaciones habituales (~/.shh) las claves del cliente.
- ***ssh_config:*** permite el análisis de un archivo de configuración OpenSSH.
- ***sock_fd:*** es un socket abierto que se utilizará para la conexión
- ***bind_addr:*** dirección IP de origen local para usar, debe ser accesible desde un dispositivo remoto.
- ***sock:*** es un socket de Python ya abierto que es utilizará para la conexión.

4. Consultar sobre el módulo xml.dom.minidom.

RESPUESTA:

Es una implementación mínima de la interfaz DOM (Document Object Mode) con una API similar a la de otros lenguajes [44]. Permite el análisis XML a través de las siguientes funciones:

- **xml.dom.minidom.parse**(*filename_or_file, parser=None, bufsize=None*): Retorna un objeto Document (Información de un documento XML) a partir de la entrada dada: *filename_or_file* que puede ser un nombre de archivo o un objeto similar a un archivo [44].
- **xml.dom.minidom.parseString**(*string, parser=None*): Retorna un objeto Document a partir de un string [44].

Objetos del DOM [44]

- node.unlink(): Funciona como un administrador de contexto para poder trabajar con los objetos Document.
- node.wirtexml(*writer*): Escribe un XML en el objeto writer. Dicho objeto solo recibe texto y no bytes como entrada.
- node.toxml(): Retorna una cadena de caracteres o una cadena de bytes que contiene el objeto Document.
- node.toprettyxml(): Retorna una versión impresa ordenada y elegante del objeto Document.

Es necesario mencionar que “node” representa un objeto Document.

ANEXO VIII

INFORME DE LA PRÁCTICA 1

a. Presentar capturas de pantalla del proceso de configuración realizado, con la debida explicación de ser necesario.
   NOTA: Algunos enunciados del procedimiento requieren respuesta o alguna evidencia de comprobación. Recuerde incluir los mismos.

   Dependerá del estudiante mostrar las capturas de pantalla necesarias para evidenciar el proceso de configuración realizado en la práctica. A continuación, se muestra las respuestas y/o evidencias que deben ser necesariamente incluidas en esta parte del informe.

   Actividad 1: Implementación del entorno de trabajo
   Sección B - Integrar GNS3 con la máquina virtual GNS3
   Literal g: De acuerdo a lo consultado en el trabajo preparatorio, ¿La configuración de la máquina virtual de GNS3 fue la adecuada para poder ejecutar la imagen de disco router Cisco CSR1000v? Argumente su respuesta.

   RESPUESTA: Fue la adecuada. La configuración de GNS3 VM sobrepasó los requerimientos mínimos de instalación de la imagen de disco del CSR1000V que son 1vCPU, 4GB RAM y 8 GB de tamaño de disco duro virtual, por lo que no existirá ningún problema al ejecutar la imagen.

   Actividad 2: Armado de topología de red y establecimiento de conexión entre dispositivos.
   Literal g: Verificar la conectividad entre la máquina virtual DEVASC-LABVM y el router Cisco CSR1000v.

   RESPUESTA: La Figura 1 muestra una prueba de ping exitosa desde la máquina virtual DEVASC-LABVM hacia el router Cisco CSR1000v, demostrando la conectividad entre ambos dispositivos.

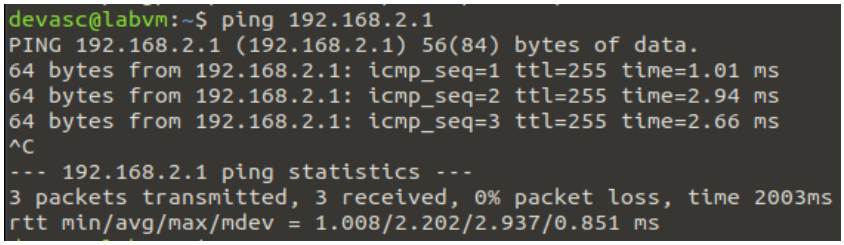


Figura 1. Prueba ping de conectividad

Actividad 3: Primeros contactos con NETCONF

Sección A - Configuración de SSH en router Cisco CSR1000v

Literal f: Abrir una terminal en la máquina virtual DEVASC y verificar el funcionamiento de SSH.

Figura 2. Funcionamiento de SSH

RESPUESTA: La Figura 2 muestra que se ha configurado con éxito SSH en el router Cisco CSR1000v.

Sección B - Habilitar servicio NETCONF en router Cisco CSR1000v

Literal c: Verificar si NETCONF ya se encuentra activo en el Agente.

RESPUESTA: NETCONF se encuentra activo en el Agente al tener habilitado el demonio NETCONF SSH (ncsshd) como se muestra en la Figura 3.

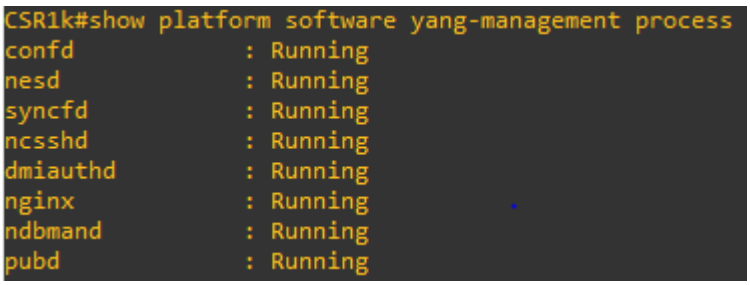


Figura 3. Demonio NETCONF SSH activo en Agente

Sección C - Comprobar funcionamiento

Literal a: Abrir una nueva terminal en la máquina virtual DEVASC-LABVM y ejecutar NETCONF como un subsistema de SSH.

RESPUESTA: El Administrador al acceder al proceso de NETCONF mediante un subsistema de SSH, deberá recibir automáticamente un mensaje <hello> que contenga las capacidades que soporta el Agente. La Figura 4 muestra la parte final del dicho mensaje.

```
<capability>urn:ietf:params:xml:ns:yang:smiv2:UDP-MIB?module=UDP-MIB&
amp;revision=2005-05-20</capability>
<capability>urn:ietf:params:xml:ns:yang:smiv2:VPN-TC-STD-MIB?module=V
PN-TC-STD-MIB&amp;revision=2005-11-15</capability>
<capability>
        urn:ietf:params:netconf:capability:notification:1.1
      </capability>
</capabilities>
<session-id>26</session-id></hello>]]>]]>
```

Figura 4. Parte del mensaje <hello> enviado por el Agente

Literal c: Verificar si la sesión de NETCONF está activa. Para ello emplear el comando show netconf-yang sessions en la terminal que inició SSH.

RESPUESTA: Una vez que el Administrador haya enviado un mensaje <hello> con al menos la capacidad base de NETCONF hacia el Agente, se establecerá una sesión NETCONF como se muestra en la Figura 5.

```
CSR1k#show netconf-yang sessions
R: Global-lock on running datastore
C: Global-lock on candidate datastore
S: Global-lock on startup datastore

Number of sessions : 1

session-id  transport    username            source-host           global-lock
-------------------------------------------------------------------------------
26          netconf-ssh  admin               192.168.2.10          None
```

Figura 5. Sesiones NETCONF activas en el Agente

Literal e: Interprete la respuesta del Agente, no es necesario ser muy técnico.

RESPUESTA: El Agente envía un mensaje <rpc-reply> que contiene un elemento <ok> que indica que se ha cerrado con éxito la sesión de NETCONF. (ver Figura 6)

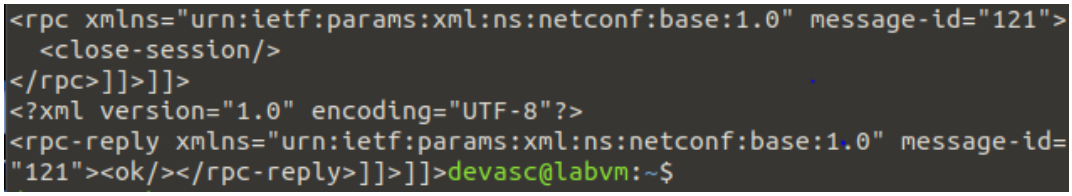


Figura 6. Intercambio de mensajes para cerrar una sesión NETCONF

Literal f: Verificar si la sesión NETCONF ha terminado

RESPUESTA: Cerrada la sesión NETCONF, no se encontrará activa ninguna sesión NETCONF en el Agente. Esto se aprecia en la Figura 7.

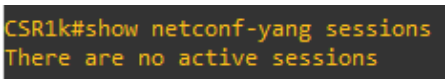


Figura 7. Sesiones NETCONF no activas en el Agente

b. De los dispositivos que se pueden emplear en GNS3 consulte sobre el router Cisco XRv 9000 y el switch Cisco NX-OSv 9000. Incluya los requerimientos de instalación para entornos KVM y los comandos para habilitar NETCONF. Colocar en formato de tabla.

Dispositivo	**Requerimientos de instalación**	**Comandos NETCONF**
Cisco XRv 9000 versión 7.5.1	- 2vCPU - 12 GB memoria RAM - 64 GB de tamaño de disco duro virtual	- Habilitar Agente NETCONF sobre SSH: `netconf-yang agent ssh` - Configurar VRF para el servidor NETCONF. `ssh server netconf` [`vrf` *nombre vrf* [`ipv4 access-list` *nombre de lista de acceso ipv4*] [`ipv6 access-list` *nombre de lista de acceso ipv6*]] - Configurar un puerto para el servidor NETCONF: `ssh server netconf port` *número de puerto*

Cisco NX-OSv 9000	- 1 a 4 vCPU - Memoria RAM de 8GB - 4 GB de tamaño de disco duro virtual	- Habilitar servicio NETCONF: `feature netconf` - (Opcional) Especifica el tiempo de espera después del cual se desconectan las sesiones de los clientes inactivos: `netconf idle-timeout` *tiempo en minutos* - (Opcional) Especifica el número máximo de sesiones de clientes simultáneos: `netconf sessions` *número de sesiones*

c. Conclusiones y Recomendaciones

Dependerá de los estudiantes escribir sus propias conclusiones y recomendaciones.

INFORME DE LA PRÁCTICA 2

a. Presentar capturas de pantalla del proceso de configuración realizado, con la debida explicación de ser necesario.
NOTA: Algunos enunciados del procedimiento requieren alguna evidencia de comprobación. Recuerde incluir los mismos.

Dependerá del estudiante mostrar las capturas de pantalla necesarias para evidenciar el proceso de configuración realizado en la práctica. A continuación, se muestra las evidencias que deben ser necesariamente incluidas en esta parte del informe.

Actividad 2: NETCONF sobre SSH con autenticación de usuario por contraseña
Sección E - Iniciar sesión NETCONF empleando la base datos de usuarios local
Literal a: Acceder al proceso de NETCONF a través de SSH.

RESPUESTA: La Figura 8 muestra que se ha conseguido con éxito acceder al proceso NETCONF con usuario admin1. Note que es requerida una contraseña para acceder al proceso.

```
devasc@labvm:~$ ssh admin1@192.168.2.1 -p 830 -s netconf
admin1@192.168.2.1's password:
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<capabilities>
<capability>urn:ietf:params:netconf:base:1.0</capability>
```

Figura 8. NETCONF sobre SSH con usuario admin1

Literal c: Verificar la sesión NETCONF y luego cerrar sesión.

RESPUESTA: Se configuró el protocolo SSH para autenticar usuarios por medio de una contraseña. El método de autenticación empleado por SSH fue AAA, el cual se configuró para que la autenticación y la autorización de los usuarios hiciera uso de la base de datos de usuario local (contenida en el Agente). El resultado de dicha configuración es la Figura 9, que muestra una sesión NETCONF activa ejecutada por el usuario admin1, que es parte de la base de datos de usuario local. El resultado obtenido demuestra que lo único que realiza el protocolo NETCONF es confiar en configuración SSH mencionada anteriormente para autenticar sus usuarios NETCONF.

```
CSR1k#show netconf-yang sessions
R: Global-lock on running datastore
C: Global-lock on candidate datastore
S: Global-lock on startup datastore

Number of sessions : 1

session-id  transport    username            source-host           global-lock
-------------------------------------------------------------------------------
21          netconf-ssh  admin1              192.168.2.10          None
```

Figura 9. Sesión NETCONF activa con usuario admin1

Sección G - Sesión NETCONF con base de datos de usuarios remota

Literal b: Acceder al proceso de NETCONF a través de SSH con usuario gns3. Comente los resultados obtenidos por el debug.

RESPUESTA: La Figura 10 muestra que se ha conseguido con éxito acceder al proceso NETCONF con usuario gns3. Note que fue requerida una contraseña para acceder al proceso.

```
devasc@labvm:~$ ssh gns3@192.168.2.1 -p 830 -s netconf
gns3@192.168.2.1's password:
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<capabilities>
<capability>urn:ietf:params:netconf:base:1.0</capability>
```

Figura 10. NETCONF sobre SSH con usuario gns3

En la dos primeras líneas de la Figura 11 se muestran las salidas obtenidas al habilitar el debug. Dichas líneas indican que se empleó la lista de métodos “default” para la autenticación de inicio de sesión y la autorización EXEC. Es importante mencionar que el servidor remoto TACACS+ se configuró como el primer método de dicha lista, por tanto, TACACS+ fue empleado para la autenticación y autorización. La última línea de la Figura 11 muestra que el usuario gns3 fue autenticado con éxito y por tanto autorizado para acceder al proceso de NETCONF mediante SSH.

```
*Jun  7 23:16:00.775: AAA/AUTHEN/LOGIN (00000000): Pick method list 'default'
*Jun  7 23:16:00.854: AAA/AUTHOR (0x0): Pick method list 'default'
*Jun  7 23:16:00.859: %DMI-5-AUTH_PASSED: R0/0: dmiauthd: User 'gns3' authenticated successful
ly from 192.168.2.10:59274 and was authorized for netconf over ssh. External groups: PRIV15
```

Figura 11. Mensajes luego de establecer NETCONF sobre SSH con usuario gns3

Literal d: Verificar la sesión NETCONF y luego cerrar sesión.

RESPUESTA: Se realizó un cambio en la configuración del método de autenticación de usuarios que emplea SSH. En vez de configurar AAA para que empleara una base de datos de usuario local para la autenticación y la autorización de los usuarios, se configuró para que dichos servicios hicieran uso de una base remota (contenida en un servidor TACACS+). El resultado de dicha configuración es la Figura 12 que muestra una sesión NETCONF activa ejecutada por el usuario gns3, que es parte de la base de datos de usuario remota. Al igual que en el resultado anterior, se demuestra que lo único que realiza el protocolo NETCONF es confiar en la configuración SSH para autenticar sus usuarios NETCONF.

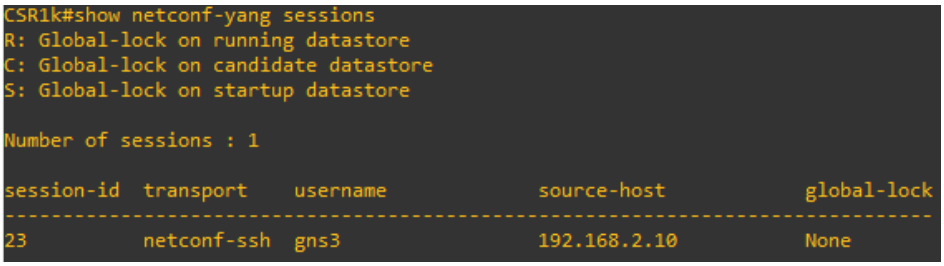


Figura 12. Sesión NETCONF activa con usuario gns3

Literal e: Apagar en AAA Docker appliance y acceder al proceso de NETCONF con usuario admin1. Comente los resultados obtenidos por el debug.

RESPUESTA: La Figura 13 muestra que se ha conseguido con éxito acceder al proceso NETCONF con usuario admin1. Note que fuere querida una contraseña para acceder al proceso.

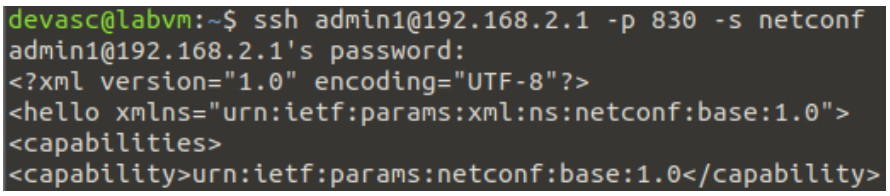


Figura 13. NETCONF sobre SSH con usuario admin1 luego de deshabilitar TACACS+

Las dos primeras líneas de la Figura 14 indican que se empleó la lista de métodos “default” para la autenticación de inicio de sesión y la autorización EXEC. Debido a que se deshabilitó el primer método de la lista (servidor remoto TACACS+), el segundo método (local) fue empleado para la autenticación y autorización. La última línea de la Figura 14 muestra que el usuario admin1 fue autenticado con éxito y por tanto autorizado para acceder al proceso de NETCONF mediante SSH.

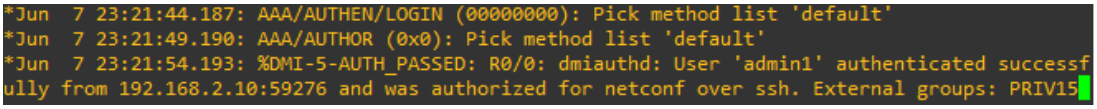


Figura 14. Mensajes luego de establecer NETCONF sobre SSH con usuario admin1

Literal f: Repita los pasos c y d

RESPUESTA: La Figura 15 muestra una sesión de NETCONF ejecutada por el usuario admin1. Se debe recalcar que dicho usuario es parte de base de datos de usuario local. Una vez más se demuestra que NETCONF se rige a la configuración de SSH para autenticar a sus usuarios NETCONF.

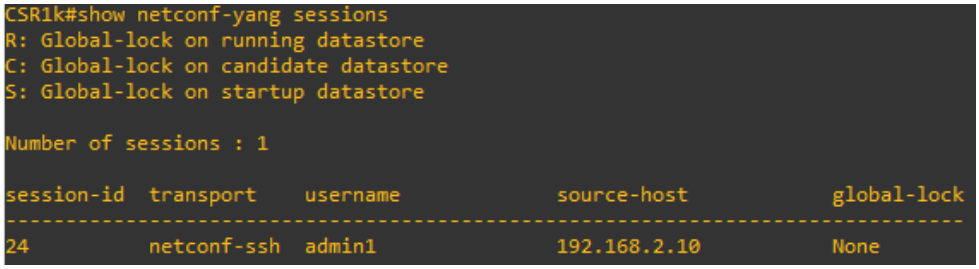


Figura 15. Sesión NETCONF activa con usuario admin1 luego de deshabilitar TACACS+

Actividad 3: NETCONF sobre SSH con autenticación de usuario por clave pública

Sección B - Configurar autenticación de usuario con clave pública

Literal g: Verificar que la clave púbica del Administrador sea la correcta.

RESPUESTA: Se aprecia que la clave pública creada en la máquina virtual DEVASC (ver Figura 16) es la misma clave asociada al usuario admin2 en el Agente (ver Figura 17). Por tanto, se verifica que la clave púbica del Administrador fue correctamente configurada en el Agente.

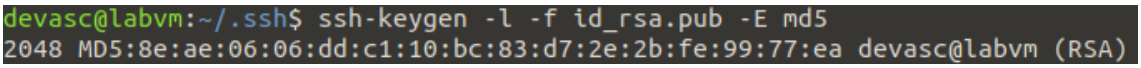


Figura 16. Clave pública del Administrador

```
ip ssh version 2
ip ssh pubkey-chain
  username admin2
   key-hash ssh-rsa 8EAE0606DDC110BC83D72E2BFE9977EA devasc@labvm
```

Figura 17. Clave publica configurada en el Agente para el usuario admin2

Sección D - Iniciar sesión NETCONF

Literal a: Acceder al proceso de NETCONF a través de SSH con usuario admin2.

RESPUESTA: La Figura 18 muestra que se consiguió con éxito acceder al proceso NETCONF con usuario admin2. Note que en la Figura 18 no se solicitó el ingreso de una contraseña.

```
devasc@labvm:~/.ssh$ ssh admin2@192.168.2.1 -p 830 -s netconf
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<capabilities>
<capability>urn:ietf:params:netconf:base:1.0</capability>
```

Figura 18. NETCONF sobre SSH con usuario admin2

Literal c: Verificar la sesión NETCONF y luego cerrar sesión.

RESPUESTA: Se configuró SSH con el método de autenticación de usuario basada en RSA. El resultado de dicha configuración es la Figura 19 que muestra una sesión NETCONF activa ejecutada por el usuario admin2, que tiene asignada la clave pública creada en el cliente (máquina virtual DEVASC). Por última vez, se demuestra que el protocolo NETCONF se rige a la configuración de SSH para autenticar a sus usuarios NETCONF.

```
CSR1k#show netconf-yang sessions
R: Global-lock on running datastore
C: Global-lock on candidate datastore
S: Global-lock on startup datastore

Number of sessions : 1

session-id  transport    username            source-host          global-lock
--------------------------------------------------------------------------------
26          netconf-ssh  admin2              192.168.2.10         None
```

Figura 19. Sesión NETCONF activa con usuario admin2

b. Realice un cuadro comparativo entre la autenticación de usuario por contraseña y por clave pública.

Contraseña	Clave pública
- Es más cómodo para los usuarios emplear contraseñas. - Es más vulnerable a ataques de fuerza bruta, por lo que no es muy segura. - Los usuarios pueden emplear contraseñas fáciles de descubrir. - Su configuración es más sencilla.	- Elimina las amenazas de los ataques de fuerza bruta. Es decir, representa una mayor seguridad. - Involucra una mayor sobrecarga en la generación y distribución de claves. - El usuario no tiene la necesidad de recordar contraseña complicadas. - Puede requerir una mayor complejidad en la configuración.

c. Describa de forma concisa las consideraciones de seguridad establecidas en el RFC 6242.

- La capacidad de acceder al proceso NETCONF debe ser limitada a usuarios y sistemas autorizados. Solo aquellos serán capaces de visualizar la configuración y el estado de un servidor NETCONF, así como modificar la configuración del mismo.
- La identidad del servidor como la identidad del cliente SSH deben ser verificadas antes de cualquier intercambio de información. Ningún lado debe establecer una conexión NETCONF sobre SSH con una identidad desconocida, inesperada o incorrecta en el lado opuesto.
- Se recomienda que los servidores SSH permitan configurar el acceso al subsistema NETCONF a través de otros puertos. El uso del puerto 830 puede permitir que los atacantes identifiquen más fácilmente el tráfico de NETCONF a través de SSH.
- El uso deliberado de la secuencia de caracteres]]>]]> puede causar problemas operativos y abrir espacio para ataques. Sin embargo, se cree que la amenaza asociada no es muy alta.

d. Conclusiones y Recomendaciones.

Dependerá de los estudiantes escribir sus propias conclusiones y recomendaciones

INFORME DE LA PRÁCTICA 3

a. Presentar capturas de pantalla del procedimiento realizado en la práctica con la debida explicación de ser necesario.
NOTA: Algunos enunciados en el procedimiento, requieren respuesta o solicitan algún análisis. Recuerde incluir los mismos

Dependerá del estudiante mostrar las capturas de pantalla necesarias para evidenciar el procedimiento realizado en la práctica. A continuación, se muestra las respuestas y/o evidencias que deben ser necesariamente incluidas en esta parte del informe

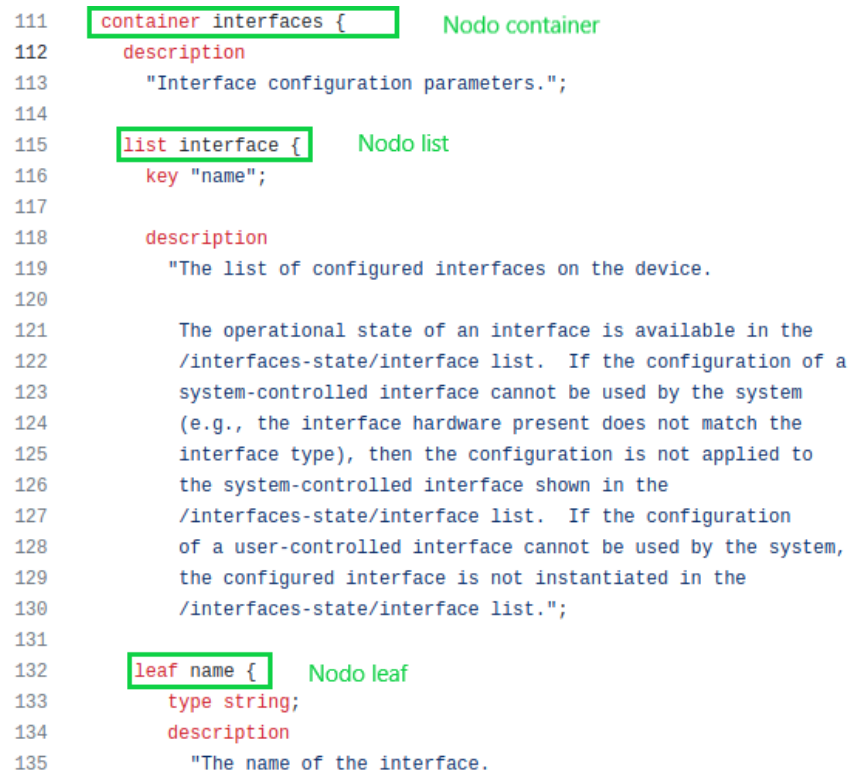
Actividad 2: Explorar modelos YANG en GITHUB
Literal d: En la lista encuentre los modelos IETF y dar clic en el ietf-interfaces.yang. Examinar todo el modelo de datos e identificar, escoger y describir un nodo `leaf`, *un* `leaf-list`, *un* `container` *y un* `list`.

La siguiente respuesta no es genérica, puede cambiar de acuerdo a lo escogido por el estudiante.
RESPUESTA:

- **Nodo interfaces**: Es un nodo del tipo `container` empleado para configurar los parámetros de las interfaces. Está compuesto de un nodo `list` denominado interface.
- **Nodo interface**: Es un nodo del tipo `list`, compuesta de varios nodos del tipo `leaf`. En si crea una secuencia de "n" listas que se identifican mediante el valor de `key`. Cada lista es una única interfaz que contiene los parámetros a configurar en la misma.
- **Nodo name**: Es un nodo del tipo `leaf`, empleado para representar el nombre de las interfaces. Dicho nombre es un dato del tipo `string`.

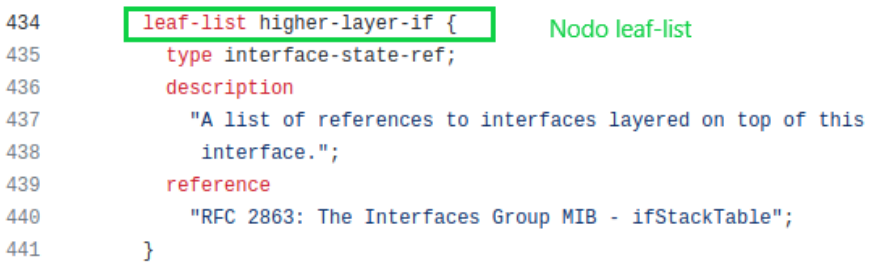
  En la Figura 20 se puede apreciar la ubicación de los nodos anteriormente mencionados. Note que no se muestran los nodos completos, debido a que poseen una gran extensión.

- **Nodo higher-layer-if**: Es un nodo del tipo `leaf-list`, que presenta un conjunto de referencias de interfaces. Todos los datos de ese conjunto son de tipo `interface-state-ref`. (ver Figura 21)




```
111   container interfaces {
112     description
113       "Interface configuration parameters.";
114
115     list interface {
116       key "name";
117
118       description
119         "The list of configured interfaces on the device.
120
121          The operational state of an interface is available in the
122          /interfaces-state/interface list.  If the configuration of a
123          system-controlled interface cannot be used by the system
124          (e.g., the interface hardware present does not match the
125          interface type), then the configuration is not applied to
126          the system-controlled interface shown in the
127          /interfaces-state/interface list.  If the configuration
128          of a user-controlled interface cannot be used by the system,
129          the configured interface is not instantiated in the
130          /interfaces-state/interface list.";
131
132       leaf name {
133         type string;
134         description
135           "The name of the interface.
```

Figura 20. Ubicación de los nodos interfaces, interface y name en *ietf-interfaces*




```
434       leaf-list higher-layer-if {
435         type interface-state-ref;
436         description
437           "A list of references to interfaces layered on top of this
438            interface.";
439         reference
440           "RFC 2863: The Interfaces Group MIB - ifStackTable";
441       }
```

Figura 21. Ubicación del nodo higher-layer-if en ietf-interfaces

Literal e: En la lista encuentre los modelos OPENCONFIG y dar clic en openconfig-interfaces.yang. Examinar todo el modelo de datos e identificar, escoger y describir un nodo `leaf`, *un* `leaf-list`, *un* `container` *y un* `list`.

La siguiente respuesta no es genérica, puede cambiar de acuerdo a lo escogido por el estudiante.

RESPUESTA:

- **Nodo interfaces**: Es un nodo del tipo `container` que incluye datos de configuración y estado de las interfaces. Está compuesto de un nodo `list` denominado interface.

- **Nodo interface**: Es un nodo del tipo `list`, que crea una secuencia de "n" listas que se identifican mediante el valor de `key`. Cada lista es una única interfaz que contiene diferentes tipos de nodos (`container`, `leaf` y `list`) que poseen datos de configuración y de estado de las interfaces.
- **Nodo name**: Es un nodo del tipo `leaf`, empleado para representar el nombre de las interfaces. Dicho nombre es un dato del tipo `leafref`.

  En la Figura 22 y Figura 20 se puede apreciar la ubicación de los nodos anteriormente mencionados. Note que dichos nodos se encuentran dentro de una declaración denominada `grouping`. Además, dichos nodos no están completos, debido a que poseen una gran extensión.

- En el modelo openconfig-interfaces no se especifica un nodo del tipo `leaf-list`.

```
932   grouping interfaces-top {
933     description
934       "Top-level grouping for interface configuration and
935       operational state data";
936
937     container interfaces {      Nodo container
938       description
939         "Top level container for interfaces, including configuration
940         and state data.";
941
942
943       list interface {          Nodo list
944         key "name";
945
946         description
947           "The list of named interfaces on the device.";
948
949         leaf name {             Nodo leaf
950           type leafref {
951             path "../config/name";
952           }
```

Figura 22. Ubicación de los nodos interfaces, interface y name en *openconfig-interfaces*

Actividad 3: Explorar modelos YANG con pyang

Literal f: Transformar los modelos de datos en formato de árbol. Seguramente notará que ambos modelos son mucho más fáciles de entender en su totalidad. Analizar y describir cada una de las salidas.

RESUESTA:

Una de las ventajas de transformar los modelos de datos en formato árbol, es que muestra de forma sencilla toda la estructura jerárquica de los datos contenidos en dichos modelos.

Al transformar el modelo *ietf-interfaces*, se aprecia que dicho modelo está conformado por dos grandes nodos del tipo `container`. El primero ofrece una estructura de datos de configuración que será empleado por los administradores en el caso de que se requiera configurar alguna interfaz. Mientras que el segundo ofrece una estructura de datos de estado. Dichos datos no podrán ser alterados y el administrador solo tendrá la posibilidad de recuperarlos. Es importante mencionar que dentro de ambos nodos del tipo `container` se encuentra un nodo del tipo `list`, y este a su vez están compuesto de otros tipos de nodos como los `leaf`, `leaf-list` y `container`.

Por otro lado, al transformar el modelo *openconfig-interfaces*, se aprecia que dicho modelo está conformado por un único nodo del tipo `container`. Dicho nodo contiene tanto las estructuras de datos de configuración como las estructuras de datos de estado de una interfaz. Es importante recalcar que dentro del nodo `container` se encuentra un nodo del tipo `list`, y este a su vez está conformado de un nodo `leaf` y cuatro nodos `container`.

Actividad 4: Ejemplo de la representación de datos de un dispositivo de red con YANG

Literal c: Analizar y comparar el ejemplo del literal anterior con el modelo de datos "ietf-interfaces". Comente lo descubierto.

RESPUESTA: A continuación, se describen los puntos más importantes al comparar el ejemplo con el modelo de datos *ietf-interfaces.*

- Los datos de configuración de las interfaces se recuperan empleando la estructura de datos contenida en el nodo `container interface` del modelo de datos.
- El módulo del modelo de datos es declarado en el elemento jerárquico de mayor nivel, que en este caso es el elemento <interfaces>.
- El elemento <interface> corresponde al nodo `container interface` del modelo de datos.
- El elemento <interfaces> posee varios elementos <interface> que corresponden a las "n" listas del nodo `list interface` del modelo de datos.
- Cada elemento <interface> es una lista bien identificada por su nombre.

- Un elemento <interface> contiene los elementos que se encuentran en el nodo `list interface` del modelo de datos. Por ejemplo, el nodo `list interface` contiene el nodo `leaf type`, por tanto, el elemento <interface> contendrá el elemento <type>.

Literal d: Responder lo siguiente: ¿A qué se debe la aparición de los nodos `container` ipv4 e ipv6, si estos no fueron descritos en el modelo "ietf-interfaces"?

RESPUESTA: La declaración `augment` es empleada en el leguaje YANG para definir la ubicación de un modelo de datos donde se insertarán los nuevos nodos que están definidos en `augment`. En el modelo de datos *ietf-ip* la declaración `augment` tiene asociado la ubicación: `/if:interfaces/if:interface` que corresponde a el nodo `list interface` del modelo de datos *ietf-interfaces.* Esto explica porque los nodos contenidos en la declaración `augment` como lo son `container` ipv4 e ipv6 son insertados en el modelo de datos *ietf-interfaces.*

Actividad 5: Identificar modelos de datos YANG que permite el Agente

Literal f: Analizar la respuesta obtenida por parte del Agente. Emplear el siguiente link: https://www.samltool.com/prettyprint.php para obtener una mejor apreciación de la información.

RESPUESTA:

El estudiante obtendrá una salida como la siguiente:

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-
id="103">
  <data>
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
      <interface>
        <name>GigabitEthernet1</name>
        <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-
type">ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
        <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
          <address>
            <ip>192.168.2.1</ip>
            <netmask>255.255.255.0</netmask>
          </address>
        </ipv4>
        <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip"/>
      </interface>
        ...[SALIDA ELIMINADA]
    </interfaces>
  </data>
```

```
</rpc-reply>
```

Al emplear la operación <get> para solicitar al Agente los datos de configuración de las interfaces. El Agente devolvió la respuesta mostrada en la parte superior, donde se aprecia que los datos de configuración siguen la estructura jerárquica definida en el modelo de datos *ietf-interfaces.*

b. Detallar las diferencias entre los modelos de datos YANG vistos en el laboratorio.

- Cada modelo de datos establece su propia jerarquización de los datos para las interfaces de un dispositivo. Es decir, son diferentes en estructura, pero tienen el mismo propósito.
- El modelo de datos *ietf-interfaces* define dos nodos `container` principales para separar los datos de configuración y de estado de las interfaces. Mientras que el modelo de datos *openconfig-interfaces* emplea un solo nodo `container` principal y es dentro del mismo donde se establecen nuevos nodos para la separación de datos de configuración y de estado de las interfaces.
- El modelo de datos *openconfig-interfaces* posee un nodo `container` que establece una jerarquía de los datos de configuración para las subinterfaces. Mientras que el modelo de datos *ietf-interfaces* no define ninguna.
- El modelo *ietf-interfaces* no reutiliza ningún grupo de nodos. Mientras que en el modelo de datos *openconfig-interfaces* se establecen varios grupos de nodos que son reutilizados constantemente en el mismo modelo de datos. En dicho modelo se emplea la declaración `grouping` para crear grupos de nodos reutilizables y se emplea la instrucción `uses` para instanciarlos.

c. Consultar y describir las declaraciones del lenguaje YANG: `typedef`, `grouping`, `uses`, `choices` y `rpc`.

`typedef`: Define un tipo derivado a partir de tipos bases. Un tipo base puede ser un tipo integrado o un tipo derivado. Es importante recalcar que un tipo derivado puede ser usado como argumento para una declaración `type`. A continuación, se muestra un ejemplo en la Figura 23.

```
typedef percent {
  type uint8 {
    range "0 .. 100";
  }
}

leaf completed {
  type percent;
}
```

Figura 23. Ejemplo YANG de la declaración `typedef`

`grouping` y **`uses`**: `grouping` permite ensamblar grupos de nodos en colecciones reutilizables. Dichas colecciones pueden ser instanciadas empleando la declaración `uses`. A continuación, se muestra un ejemplo en la Figura 24.

```
YANG Example:

  grouping target {
    leaf address {
      type inet:ip-address;
      description "Target IP address.";
    }
    leaf port {
      type inet:port-number;
       description "Target port number.";
    }
  }

  container peer {
    container destination {
      uses target;
    }
  }
```

Figura 24. Ejemplo YANG de la declaración `grouping` y `uses`

`choices`: Permite dividir nodos incompatibles en opciones distintas. Una declaración `choices` contiene un conjunto de declaraciones `case` que define un conjunto de nodos. Es importante recalcar que un solo `case` puede ser escogido para ser validado, mientras que los demás se eliminarán implícitamente. A continuación, se muestra un ejemplo en la Figura 25.

```
container food {
  choice snack {
    case sports-arena {
      leaf pretzel {
        type empty;
      }
      leaf beer {
        type empty;
      }
    }
    case late-night {
      leaf chocolate {
        type enumeration {
          enum dark;
          enum milk;
          enum first-available;
        }
      }
    }
  }
}
```

Figura 25. Ejemplo YANG de la declaración `choices`

`rpc`: Permite la creación de operaciones. Los nombres de las operaciones, los parámetros de entrada y salida se modelan empleando el lenguaje YANG. A continuación, se muestra un ejemplo en la Figura 26.

```
rpc activate-software-image {
  input {
    leaf image-name {
      type string;
    }
  }
  output {
    leaf status {
      type string;
    }
  }
}
```

Figura 26. Ejemplo YANG de la declaración `rpc`

d. Conclusiones y Recomendaciones**.**

Dependerá de los estudiantes escribir sus propias conclusiones y recomendaciones

INFORME DE LA PRÁCTICA 4

a. Presentar capturas de pantalla del desarrollo de la práctica con la debida explicación de ser necesario.

NOTA: Algunos enunciados de la práctica requieren respuesta o solicitan algún análisis. Recuerde incluir los mismos.

Dependerá del estudiante mostrar las capturas de pantalla necesarias para evidenciar el procedimiento realizado en la práctica. A continuación, se muestra las respuestas y/o evidencias que deben ser necesariamente incluidas en esta parte del informe

Actividad 3: **Enviar mensajes RPC al Agente empleando las operaciones base de NETCONF**

Literal c: Analizar el mensaje de respuesta del Agente <rpc-reply> después de solicitar la operación <get>.

RESPUESTA: El siguiente texto XML muestra la información devuelta por el Agente luego de solicitar la operación <get>. Se pude apreciar que tanto la información de configuración y de estado referente a la interfaz Gigabit Ethernet 1 fue devuelta con éxito. Es importante recalcar que dicha información es la contenida en el almacén de datos de configuración <running>.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"    message-
id="103">
  <data>
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
      <interface>
        <name>GigabitEthernet1</name>
        <type          xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-
type">ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
        <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
          <address>
            <ip>192.168.2.1</ip>
            <netmask>255.255.255.0</netmask>
          </address>
        </ipv4>
        <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip"/>
      </interface>
    </interfaces>
    <interfaces-state            xmlns="urn:ietf:params:xml:ns:yang:ietf-
interfaces">
      <interface>
        <name>GigabitEthernet1</name>
```

```
        <type          xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-
type">ianaift:ethernetCsmacd</type>
        <admin-status>up</admin-status>
        <oper-status>up</oper-status>
        <last-change>2022-04-12T02:47:42.404+00:00</last-change>
        <if-index>1</if-index>
        <phys-address>0c:d6:0f:13:00:00</phys-address>
        <speed>1000000000</speed>
        <statistics>
          <discontinuity-time>2022-04-12T02:44:37+00:00</discontinuity-
time>
          <in-octets>66125</in-octets>
          <in-unicast-pkts>339</in-unicast-pkts>
          <in-broadcast-pkts>0</in-broadcast-pkts>
          <in-multicast-pkts>0</in-multicast-pkts>
          <in-discards>0</in-discards>
          <in-errors>0</in-errors>
          <in-unknown-protos>0</in-unknown-protos>
          <out-octets>189914</out-octets>
          <out-unicast-pkts>410</out-unicast-pkts>
          <out-broadcast-pkts>0</out-broadcast-pkts>
          <out-multicast-pkts>0</out-multicast-pkts>
          <out-discards>0</out-discards>
          <out-errors>0</out-errors>
        </statistics>
      </interface>
    </interfaces-state>
  </data>
</rpc-reply>]]>]]>
```

Literal e: Analizar el mensaje de respuesta del Agente <rpc-reply> después de solicitar la operación <get-config>.

RESPUESTA: El siguiente texto XML muestra la información devuelta por el Agente luego de solicitar la operación <get-config>. A pesar de que se solicitó los datos de estado y los datos de configuración de la interfaz Gigabit Ethernet 1, dicha operación solo es capaz de devolver los datos de configuración. Es importante recalcar que la información devuelta es la contenida en el almacén de datos de configuración <candidate>. La información de la interfaz Gigabit Ethernet 1 es la misma que está contenida en el almacén de datos <running>, esto debido a que cuando se habilitó el almacén de datos <candidate> la configuración activa de <running> fue copiada directamente en el almacén <candidate>.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-
id="104">
  <data>
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
      <interface>
```

```
        <name>GigabitEthernet1</name>
        <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-
type">ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
        <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
          <address>
            <ip>192.168.2.1</ip>
            <netmask>255.255.255.0</netmask>
          </address>
        </ipv4>
        <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip"/>
      </interface>
    </interfaces>
  </data>
</rpc-reply>]]>]]>
```

Literal g: Determinar porque no se incluyó los elementos <test-option> y <error-option> en el ejemplo del literal anterior. Recuerde que no incluir algo, no siempre significa que no esté funcionando.

RESPUESTA: No es necesario incluir los valores por defecto de los elementos test-option> y <error-option>. Aunque estos no hayan sido incluidos dentro de la operación <edit-config>, los elementos <test-option> y <error-option> están configurados con sus valores por defecto *test-then-set* y *stop-on-error* respectivamente. Se debe recalcar que dichos parámetros opciones permitirán mejorar el rendimiento de <edit-config> en el caso de usarlos.

Literal h: Analizar el mensaje de respuesta del Agente <rpc-reply> después de solicitar la operación <edit-config>.

RESPUESTA: La operación <edit-config> permita realizar cualquier cambio en configuración dependiendo de su modo de operación. El siguiente texto XML muestra que se realizó con éxito la operación <edit-config>. Es decir que se logró configurar la interfaz Gigabit Ethernet 3 en el almacén de datos de configuración <candidate>.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-
id="114">
  <ok/>
</rpc-reply>]]>]]>
```

Literal j: Analizar el mensaje de respuesta del Agente <rpc-reply> después de solicitar la operación <copy-config> y luego contestar la siguiente pregunta: ¿Fue posible copiar

la configuración del almacén de datos <candidate> al almacén de datos <running>? Justifique su respuesta.

RESPUESTA: El siguiente texto XML muestra que no se realizó con éxito la operación <copy-config>. El mensaje <rpc-reply> contiene el elemento <rpc-error> que contiene información sobre el error ocurrido durante la ejecución de la operación <copy-config>. De ese mensaje se destaca el elemento <error-info> que indica que no es posible emplear el valor <running>.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-
id="106">
  <rpc-error>
    <error-type>protocol</error-type>
    <error-tag>invalid-value</error-tag>
    <error-severity>error</error-severity>
    <error-message xml:lang="en">Unsupported capability :writable-
running</error-message>
    <error-info>
      <bad-element>running</bad-element>
    </error-info>
  </rpc-error>
</rpc-reply>]]>]]>
```

Respondiendo a la pregunta, no se logró copiar la configuración del almacén de datos <candidate> al almacén de datos <running>. Esto debido a que el Agente no anunció la capacidad Writable-running. Dicha capacidad es la que permite realizar configuraciones directas en el almacén de datos <running>, es decir, es la que permite el uso de las operaciones <edit-config> y <copy-config> donde <running> es el destino.

Literal l: Analizar el mensaje de respuesta del Agente <rpc-reply> después de solicitar la operación <delete-config> y luego contestar la siguiente pregunta: ¿Fue posible eliminar la configuración del almacén de datos <running>? Justifique su respuesta.

RESPUESTA: El siguiente texto XML muestra que no se realizó con éxito la operación <delete-config>. El mensaje <rpc-reply> contiene el elemento <rpc-error> que contiene información sobre el error ocurrido durante la ejecución de la operación. Se destaca el elemento <error-path> que indica la ruta donde se produjo el error y el elemento <error-info> que indica que no es posible emplear el valor <running>.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-
id="107">
```

```
  <rpc-error>
    <error-type>protocol</error-type>
    <error-tag>unknown-element</error-tag>
    <error-severity>error</error-severity>
    <error-path>
    /rpc/delete-config/target
  </error-path>
    <error-info>
      <bad-element>running</bad-element>
    </error-info>
  </rpc-error>
</rpc-reply>]]>]]>
```

Respondiendo a la pregunta, no fue posible eliminar el almacén de datos <running>. Según las especificaciones de la operación <delete-config>, el único almacén de datos que no puede ser eliminado en un servidor NETCONF es el almacén <running>.

Literal o: Analizar los mensajes de respuesta del Agente <rpc-reply> después de solicitar la operación <lock>.

RESPUESTA: Los siguientes textos XML muestra que se bloqueó con éxito los almacenes de datos <running> y <candidate>.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-
id="108">
  <ok/>
</rpc-reply>]]>]]>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-
id="109">
  <ok/>
</rpc-reply>]]>]]>
```

Literal p: Iniciar sesión NETCONF en otra terminal e intentar configurar la interfaz Gigabit Ethernet 3 del almacén de datos <candidate>. También ingresar a la CLI del router e intentar realizar la misma configuración. Tenga en cuenta que al usar el CLI está configurando sobre el almacén de datos <running>. Comente los resultados.

RESPUESTA: Al verificar las sesiones NETCONF activas en el Agente (ver Figura 27), se puede evidenciar que en la sesión 28 se han bloqueado los almacenes de configuración <candiate> y <running>. Mientras que en la sesión 29 no existe ningún almacén bloqueado.

```
CSR1k#show netconf-yang sessions
R: Global-lock on running datastore
C: Global-lock on candidate datastore
S: Global-lock on startup datastore

Number of sessions : 2

session-id  transport    username            source-host           global-lock
-----------------------------------------------------------------------------
28          netconf-ssh  admin2              192.168.2.10          R, C
29          netconf-ssh  admin2              192.168.2.10          None
```

Figura 27. Sesiones NETCONF activas luego de la operación <lock>

Se utilizó la sesión 29, para enviar al Agente una operación <edit-config> solicitando configurar la interfaz Gigabit Ethernet 3 en el almacén de datos <candidate>. La respuesta del Agente se evidencia en el siguiente texto XML. Note que en el mensaje <rpc-reply> el elemento <rpc-error> indica que se ha producido un error. No fue posible la configuración debido a que la sesión 28 ha bloqueado el almacén de datos <candidate>.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-
id="103">
  <rpc-error>
    <error-type>application</error-type>
    <error-tag>in-use</error-tag>
    <error-severity>error</error-severity>
    <error-app-tag>config-locked</error-app-tag>
    <error-info>
      <session-id>28</session-id>
    </error-info>
  </rpc-error>
</rpc-reply>]]>]]>
```

Por otro lado, al emplear la CLI para configurar la interfaz Gigabit Ethernet 3 del almacén de datos <running>, se obtuvo el mensaje mostrado en la Figura 28. Dicho mensaje indica que no fue posible entrar a el modo de configuración debido a que un usuario NETCONF ha bloqueado el acceso al mismo.

```
CSR1k#conf te
Configuration mode is locked by process '342' user 'NETCONF' from terminal '64'. Please try later.
```

Figura 28. Mensaje al intentar acceder al modo de configuración

No importa qué tipo de cliente (NETCONF o no) intente realizar alguna configuración sobre un almacén de datos que ha sido bloqueado, de igual forma no lo conseguirá.

Literal r: Analizar los mensajes de respuesta del Agente <rpc-reply> después de solicitar la operación <unock> y repita el literal n.

RESPUESTA: La sesión que realizo el bloqueo de los almacenes de datos, es la única que puede desbloquear los mismos. Los siguientes textos XML muestra que se desbloqueó con éxito los almacenes de datos <running> y <candidate>.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-
id="108">
  <ok/>
</rpc-reply>]]>]]>

<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-
id="109">
  <ok/>
</rpc-reply>]]>]]>
```

Al verificar las sesiones NETCONF activas en el Agente (ver Figura 29), se puede evidenciar que ninguna sesión tiene bloqueado un almacén de datos de configuración.

```
CSR1k#show netconf-yang sessions
R: Global-lock on running datastore
C: Global-lock on candidate datastore
S: Global-lock on startup datastore

Number of sessions : 2

session-id  transport    username            source-host           global-lock
--------------------------------------------------------------------------------
28          netconf-ssh  admin2              192.168.2.10          None
29          netconf-ssh  admin2              192.168.2.10          None
```

Figura 29. Sesiones NETCONF activas luego de la operación <unlock>

Se utilizó la sesión 29, para enviar al Agente una operación <edit-config> solicitando configurar la interfaz Gigabit Ethernet 3 en el almacén de datos <candidate>. La respuesta del Agente se evidencia en el siguiente texto XML. Note que el mensaje <rpc> muestra que se consiguió con éxito la configuración.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-
id="104">
  <ok/>
</rpc-reply>]]>]]>
```

Por otro lado, al emplear la CLI ya fue posible entrar el modo de configuración. (ver Figura 30)

```
CSR1k#conf ter
Enter configuration commands, one per line.  End with CNTL/Z.
CSR1k(config)#
```

Figura 30. Acceso al modo de configuración

Literal u: Iniciar sesión NETCONF en dos terminales distintas y luego usar el comando <kill-session> en la primera terminal para forzar la terminación de la sesión NETCONF de la segunda terminal. ¿Fue posible? Justique su respuesta.

RESPUESTA: La Figura 31 muestra que se encuentran activas dos sesiones NETCONF (la 30 y la 31).

```
CSR1k#show netconf-yang sessions
R: Global-lock on running datastore
C: Global-lock on candidate datastore
S: Global-lock on startup datastore

Number of sessions : 2

session-id  transport    username            source-host           global-lock
--------------------------------------------------------------------------------
30          netconf-ssh  admin2              192.168.2.10          None
31          netconf-ssh  admin2              192.168.2.10          None
```

Figura 31. Sesiones NETCONF activas antes de <kill-session>

En la sesión 31 se empleó la operación <kill-session> para forzar el cierre de la sesión 30. Para ello se utilizó el siguiente texto XML.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="113">
  <kill-session>
    <session-id>30</session-id>
  </kill-session>
</rpc>]]>]]>
```

La Figura 32 muestra que se ha conseguido con éxito el cierre forzado de la sesión 30.

```
CSR1k#show netconf-yang sessions
R: Global-lock on running datastore
C: Global-lock on candidate datastore
S: Global-lock on startup datastore

Number of sessions : 1

session-id  transport    username            source-host           global-lock
--------------------------------------------------------------------------------
31          netconf-ssh  admin2              192.168.2.10          None
```

Figura 32. Sesiones NETCONF activas despues de <kill-session>

Actividad 4: **Operaciones adicionales permitidas por las capacidades anunciadas en el Agente**

Literal d: Añadir una PC a la topología, configurar la dirección 192.168.4.10/24 y conectar la misma al puerto Gigabit Ethernet 3 del router. ¿Se logró establecer la conexión? Justifique su respuesta.

RESPUESTA: Si se logró establecer la conexión entre la PC y el router (ver Figura 33). Al emplear <commit> la configuración contenida en el almacén de datos de <candidate> pasó a ejecutarse como la configuración de ejecución del dispositivo. Se debe recordar que la interfaz Gigabit Ethernet 3 fue configurada con la dirección 192.168.4.1/24 en el almacén de datos <candidate>, razón por la cual fue posible la conexión entre ambos dispositivos.

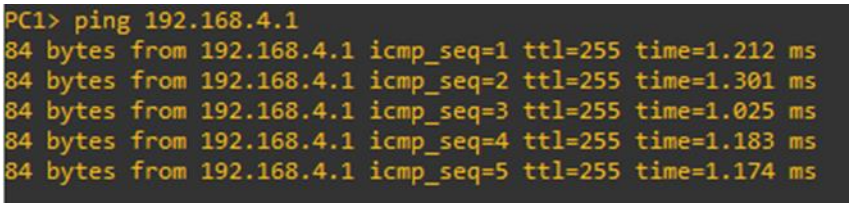


Figura 33. Prueba ping desde PC a Router

Literal f: Analizar el mensaje de respuesta del Agente <rpc-reply> después de solicitar la operación <discard-changes > y repita el literal d. Comente el resultado.

RESPUESTA: El siguiente texto XML muestra que la operación <discard-changes> fue ejecutada con éxito.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-
id="115">
  <ok/>
</rpc-reply>]]>]]>
```

Aún se mantiene la conexión entre PC y router. La operación <discard-changes> no tendrá ningún efecto sobre el almacén de datos <candidate> que primero fue empleado en la operación <commit>.

Literal j: Verificar la conectividad entre PC y router, y luego de dos minutos intentar realizar la misma prueba de conectividad. Comente los resultados y luego responda ¿Qué significa confirmar <commit>?

RESPUESTA: Al emplear la operación <commit> con el parámetro <confirmed>, la configuración contenida en el almacén de datos de <candidate> se mantendrá activa como la configuración de ejecución del dispositivo por un tiempo límite, en este caso fue 2 minutos (ver Figura 34 y Figura 35)

En respuesta a la pregunta. Confirmar <commit> significa renovar la operación <commit> antes de un tiempo establecido, caso contrario <commit> se anulará.

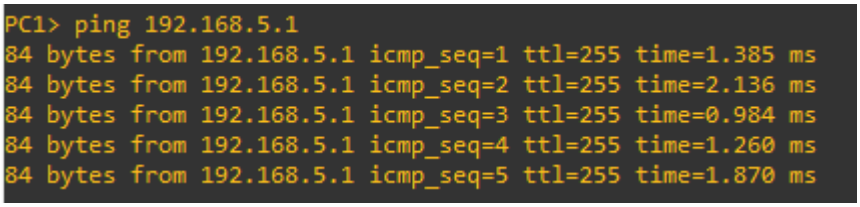


Figura 34. Conexión entre PC y router durante 2 minutos

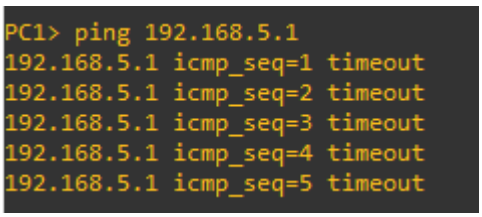


Figura 35. Conexión entre PC y router luego de 2 minutos

Literal m: Emplear la operación <cancel-commit> para desactivar la confirmación de <commit> y luego volver realizar la prueba de conectividad. Comente los resultados.

RESPUESTA: La operación <cancel-commit> permite cancelar un <commit> que necesita confirmación. En la Figura 36 se muestra que se detuvo la operación <commit> antes de llegar al tiempo límite de 5 minutos.

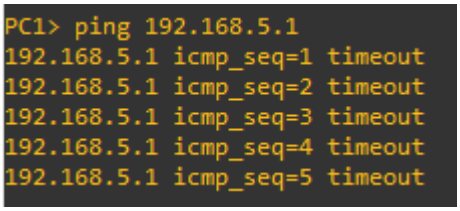


Figura 36. Conexión entre PC y router luego de <cancel-commit>

Literal o: Analizar el mensaje de respuesta del Agente <rpc-reply> después de solicitar la operación <validate>.

RESPUESTA: Una de las funcionalidades de la operación <validate> es que permite verificar el contenido de un almacén de datos de configuración especifico. El siguiente texto XML muestra que no existen errores en el almacén de datos <candidate>.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-
id="120">
  <ok/>
</rpc-reply>]]>]]>
```

b. Emplear la operación base <edit-config> para eliminar la interfaz Gigabit Ethernet 4 del almacén de datos <candidate> que fue creada en el desarrollo de la práctica. Compruebe los resultados.

NOTA: Recuerde que el atributo “operation” con el valor “delete”, permite eliminar parte de la configuración.

Se empleó el siguiente texto XML para eliminar la interfaz Gigabit Ethernet 4 del almacén de datos <candidate>.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="105">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
      <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
        <interface xc:operation="delete">
          <name>GigabitEthernet4</name>
        </interface>
      </interfaces>
    </config>
  </edit-config>
</rpc>
```

Se verificó la correcta eliminación de la interfaz Gigabit Ethernet 4, al evaluar la respuesta que produjo la siguiente solicitud enviada hacia el Agente. Note que dicha solicitud pide la información de configuración de la interfaz Gigabit Ethernet 4.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="104">
  <get-config>
    <source>
      <candidate/>
    </source>
    <filter>
      <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
        <interface>
          <name>GigabitEthernet4</name>
        </interface>
      </interfaces>
```

```
    </filter>
  </get-config>
</rpc>]]>]]>
```

La respuesta del Agente se evidencia en el siguiente texto XML. Note que este no posee ningún dato referente a la interfaz Gigabit Ethernet 4 del almacén <candidate>, lo que significa que se ha eliminado con éxito dicha interfaz.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-
id="104">
  <data/>
</rpc-reply>]]>]]>
```

c. Conclusiones y Recomendaciones

Dependerá de los estudiantes escribir sus propias conclusiones y recomendaciones

INFORME DE LA PRÁCTICA 5

a. Presentar capturas de pantalla del desarrollo de la práctica con la debida explicación de ser necesario.

   NOTA: Algunos enunciados de la práctica requieren respuesta o solicitan algún análisis. Recuerde incluir los mismos.

   Dependerá del estudiante mostrar las capturas de pantalla necesarias para evidenciar el procedimiento realizado en la práctica. A continuación, se muestra las respuestas y/o evidencias que deben ser necesariamente incluidas en esta parte del informe

   Actividad 2: **Inicio de sesión NETCONF en Agente con ncclient**

   Literal h: Comprobar que el Agente aceptó el requerimiento para una sesión NETCONF, para ello visualizar el mensaje %DMI-5-AUTH_PASSED generado en la consola del dispositivo. Recuerde que debe comprobar tanto la conexión SSH con contraseña como la de clave pública. Para ello deberá comentar parte del código en ncclient-connection-netconf.py

   RESPUESTA: La Figura 37 muestra los mensajes %DMI-5-AUTH_PASSED obtenidos en la CLI del CSRk1, luego de aceptar el requerimiento para una sesión NETCONF con dos métodos de autenticación: por contraseña y por clave pública. Se debe destacar que se empleó el usuario admin1 para la conexión NETCONF con la autenticación por contraseña y el usuario admin2 para la autenticación por clave pública.

```
*Apr 16 22:07:12.829: %DMI-5-AUTH_PASSED: R0/0: dmiauthd: User 'admin2' authenticated successfully from 192.168.2.10:60656 a
nd was authorized for netconf over ssh. External groups: PRIV15
*Apr 16 23:34:21.689: %DMI-5-AUTH_PASSED: R0/0: dmiauthd: User 'admin1' authenticated successfully from 192.168.2.10:60660 a
nd was authorized for netconf over ssh. External groups: PRIV15
```

Figura 37. Mensajes %DMI-5-AUTH_PASSED

   Literal k: Guardar y probar el programa. Comente el resultado.

   RESPUESTA: Al ejecutar el programa, se obtiene la lista de capacidades que soporta el Agente. Parte de esa lista se muestra en la Figura 38.

Figura 38. Salida obtenida al ejecutar el programa *ncclient-server-capabilities.py*

Actividad 3: **Recuperación de información con ncclient**

Literal c: Guardar y probar el programa. Comente el resultado. Puede usar XML Pretty Print Online Tool para visualizar mejor los datos recuperados.

RESPUESTA: Al ejecutar el programa, se obtuvo la configuración en ejecución contenida en el Agente. En la Figura 39 se puede apreciar parte de dicha información, la cual resulta difícil de comprender.

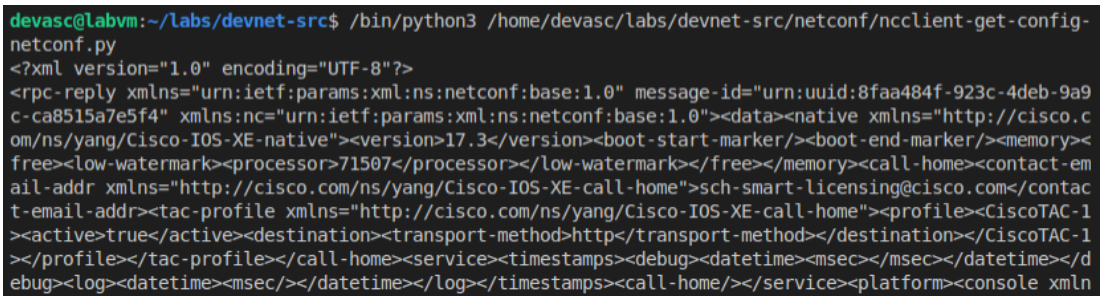


Figura 39. Salida obtenida al ejecutar el programa *ncclient-get-config-netconf.py*

Se empleó XML Pretty Print Online Tool para facilitar la visualización de los datos enviados por el Agente. De la información recibida se destaca que la estructura jerárquica que siguen los datos recibidos es la establecida en el modelo de datos Cisco-IOS-XE-native. El siguiente texto muestra parte de dicha información luego de emplear XML Pretty Print Online Tool.

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" message-
id="urn:uuid:8faa484f-923c-4deb-9a9c-ca8515a7e5f4">
  <data>
    <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
      <version>17.3</version>
      <boot-start-marker/>
      <boot-end-marker/>
      <memory>
```

```
        <free>
          <low-watermark>
            <processor>71507</processor>
          </low-watermark>
        </free>
      </memory>
      <call-home>
        <contact-email-addr xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-
call-home">sch-smart-licensing@cisco.com</contact-email-addr>
        <tac-profile xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-call-
home">
          <profile>
            <CiscoTAC-1>
              <active>true</active>
              <destination>
                <transport-method>http</transport-method>
              </destination>
            </CiscoTAC-1>
          </profile>
        </tac-profile>
      </call-home>
      <service>
        <timestamps>
          <debug>
            <datetime>
              <msec/>
            </datetime>
          </debug>
          <log>
            <datetime>
              <msec/>
            </datetime>
          </log>
        </timestamps>
        <call-home/>
      </service>
      <platform>
        <console xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-platform">
          <output>serial</output>
        </console>
        <qfp xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-platform">
          <utilization>
            <monitor>
              <load>80</load>
            </monitor>
          </utilization>
        </qfp>
        <punt-keepalive xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-
platform">
          <disable-kernel-core>true</disable-kernel-core>
        </punt-keepalive>
      </platform>
      <hostname>CSR1k</hostname>
      <username>
        <name>admin</name>
        <privilege>15</privilege>
```

```
      <password>
        <encryption>0</encryption>
        <password>c1sco12345</password>
      </password>
    </username>
    <username>
      <name>admin1</name>
      <privilege>15</privilege>
      … [SALIDA ELIMINADA]
```

Literal e: Guardar y probar el programa. Comente el resultado

RESPUESTA: La Figura 40 muestra la salida obtenida luego de la modificación expuesta en el literal d. Gracias a los métodos que ofrece el módulo *xml.dom.minidom,* los datos devueltos por el Agente en formato XML se presentan de una manera más ordenada, permitiendo una mejor apreciación de los mismos. Ya no existe la necesidad de usar XML Pretty Print Online Tool para visualizar los datos devueltos por el Agente.

Figura 40. Salida obtenida al ejecutar el programa *ncclient-get-config-netconf.py* modificado

Literal i: Guardar y probar el programa. Comente el resultado

RESPUESTA: La Figura 41 muestra la salida obtenida al ejecutar el programa *ncclient-get-config-int-netconf.py.* Dicho resultado se debe principalmente al uso del parámetro filter en el método get_config(). Gracias a dicho parámetro es posible extraer parte de una configuración de un almacén de datos disponible en el Agente. Es importante recalcar que filter tiene asociado una variable que es la que contiene la jerarquía de datos (de configuración o de estado) la cual es específica de un modelo de datos.

Figura 41. Salida obtenida al ejecutar el programa *ncclient-get-config-int-netconf.py*

Actividad 4: **Configuración del Agente con ncclient**

Literal d: Guardar y probar el programa. Comente el resultado.

RESPUESTA: La Figura 42 muestra la salida obtenida al ejecutar el programa *ncclient-edit-config-netconf.py.* El resultado obtenido que se observa en dicha figura indica que se ha conseguido con éxito realizar la configuración solicitada. Es importante recalcar que se empleó el parámetro config en el método edit_config(), ya que el mismo es el que tiene asociado una variable que contine la estructura de datos a configurar, la cual es específica de un modelo de datos.

Figura 42. Salida obtenida al ejecutar el programa *ncclient-edit-config-netconf.py*

Literal g: Guardar y probar el programa del Código 11. Comente los resultados obtenidos.

RESPUESTA: Al ejecutar el programa del Código 11, lo primero que se solicitó al administrador fue ingresar por consola la información para configurar una nueva interfaz loopback (ver Figura 43). Si los datos fueron ingresados con éxito, el programa retornará la salida mostrada en la Figura 44, indicando que se ha conseguido con éxito la configuración de la nueva interfaz.

```
devasc@labvm:~/labs/devnet-src$ /bin/python3 /home/devasc/labs/devnet-src/netconf/ncclient-new-loopback-netconf.py
Configuración de Loopback
Número de la interfaz Loopback a crear: 1
Descripción de la interfaz Loopback: Mi primera NETCONF Loopback
Dirección IP: 2.2.2.2
Mascara de red: 255.255.255.255
```

Figura 43. Ingreso de parámetros para la configuración de la interfaz loopback

```
<?xml version="1.0" ?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" messag
e-id="urn:uuid:939c4563-6b51-4c99-accc-93f601f1eb39">
       <ok/>
</rpc-reply>

devasc@labvm:~/labs/devnet-src$
```

Figura 44. Salida del programa luego de ingresar los parámetros de la interfaz loopback

La Figura 45 muestra a manera de comprobación adicional la correcta creación de la interfaz loopback. Se debe mencionar que se empleó una variación del programa *ncclient-get-config-netconf.py* para obtener dicha respuesta.

```
                <negotiation xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-ethernet">
                        <auto>true</auto>
                </negotiation>
        </GigabitEthernet>
        <Loopback>
                <name>1</name>
                <description>Mi primera NETCONF Loopback</description>
                <ip>
                        <address>
                                <primary>
                                        <address>2.2.2.2</address>
                                        <mask>255.255.255.255</mask>
                                </primary>
                        </address>
                </ip>
        </Loopback>
</interface>
```

Figura 45. Interfaz loopback creada

Actividad 5: **Eliminación de información de configuración con ncclient**

Literal c: Guardar y probar el programa ncclient-delete-loopback-netconf.py. Comente el resultado.

RESPUESTA: Al ejecutar el programa, lo primero que se solicitó al administrador fue ingresar el número de la interfaz loopback a eliminar. En el caso de haber ingresado el valor correcto aparecerá un mensaje <rpc> con el elemento <ok> indicando que se ha eliminado con éxito dicha interfaz (ver Figura 46).

```
devasc@labvm:~/labs/devnet-src$ /bin/python3 /home/devasc/labs/devnet-src/netconf/ncclient-delete-loopback-netconf.py
Configuración de Loopback
Número de la interfaz Loopback a eliminar: 1
<?xml version="1.0" ?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:a29f573d-4b68-45d5-8113-6a035d8b1e3f">
        <ok/>
</rpc-reply>
```

Figura 46. Salida obtenida al ejecutar el programa *ncclient-delete-loopback-netconf.py*

La Figura 47 muestra a manera de comprobación adicional la correcta eliminación de la interfaz Loopback.

```
            <GigabitEthernet>
                    <name>4</name>
                    <shutdown/>
                    <logging>
                            <event>
                                    <link-status/>
                            </event>
                    </logging>
                    <mop>
                            <enabled>false</enabled>
                            <sysid>false</sysid>
                    </mop>
                    <negotiation xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-ethernet">
                            <auto>true</auto>
                    </negotiation>
            </GigabitEthernet>
    </interface>
```

Figura 47. Interfaz Loopback eliminada

Literal f: Guardar y probar el programa creado. Comente el resultado.

RESPUESTA: En la Figura 48 se muestra la salida después de ejecutar el programa *ncclient-delete-config-netconf.py*. La respuesta obtenida indica que no se ha realizado con éxito la configuración solicitada, es decir no se logró eliminar el almacén de datos <running>. Se debe recordar que <running> es el único almacén que no puede ser eliminado con la operación <delete-config>.

```
devasc@labvm:~/labs/devnet-src$ /bin/python3 /home/devasc/labs/devnet-src/netconf/ncclient-delete-config-netconf.py
Traceback (most recent call last):
  File "/home/devasc/labs/devnet-src/netconf/ncclient-delete-config-netconf.py", line 36, in <module>
    netconf_reply = m.delete_config(target="running")
  File "/home/devasc/.local/lib/python3.8/site-packages/ncclient/manager.py", line 246, in execute
    return cls(self._session,
  File "/home/devasc/.local/lib/python3.8/site-packages/ncclient/operations/edit.py", line 90, in request
    return self._request(node)
  File "/home/devasc/.local/lib/python3.8/site-packages/ncclient/operations/rpc.py", line 375, in _request
    raise self._reply.error
ncclient.operations.rpc.RPCError: {'type': 'protocol', 'tag': 'unknown-element', 'app_tag': None, 'severity': 'error',
'info': '<?xml version="1.0" encoding="UTF-8"?><error-info xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"><bad-element>running</bad-element>\n</error-info>\n', 'path': '\n    /rpc/delete-config/target\n  ', 'message': None}
```

Figura 48. Salida obtenida al ejecutar el programa *ncclient-delete-config-netconf.py*

Actividad 6: **Guardar configuración del Agente con ncclient**

*Literal f: Guardar y probar el programa. Comente el resultado***.**

RESPUESTA: Es necesario mencionar que el programa, primero realiza una operación <commit> y luego hace uso de <save-config>. En otras palabras, primero establece la configuración del almacén <candidate> como la configuración en ejecución del dispositivo y luego la copia en la configuración de inicio del Agente.

Se debe recordar que el mensaje del día fue configurado en <candidate> en la Actividad 4. Debido a que se obtuvo una respuesta positiva al ejecutar el programa, el mensaje del día debería estar presente en la configuración de inicio. Esto se verifica en la Figura 50.

```
devasc@labvm:~/labs/devnet-src$ /bin/python3 /home/devasc/labs/devnet-src/netconf/ncclient-save-config-netconf.py
<?xml version="1.0" ?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" mes
sage-id="urn:uuid:4bab5b19-d0ec-40ba-909c-816f09ad0ca6">
        <result xmlns="http://cisco.com/yang/cisco-ia">Save running-config successful</result>


</rpc-reply>
```

Figura 49. Salida obtenida al ejecutar el programa *ncclient-save-config-netconf.py*

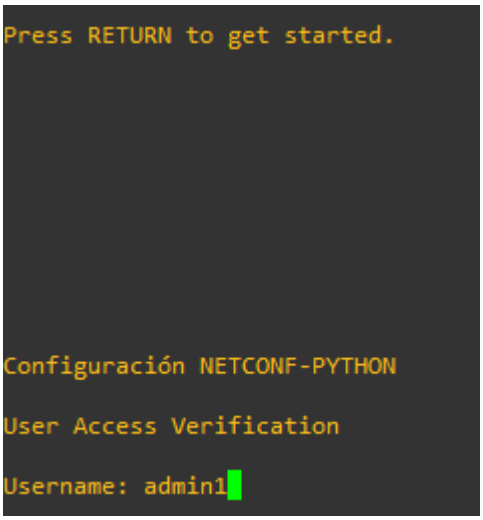


Figura 50. Mensaje del día presente al iniciar el Agente

b. Configure OSPF usando ncclient. Tenga en cuenta la topología de red de la Figura 51
NOTA: Debe emplear el modelo de datos Cisco-IOS-XE-ospf, específicamente enfocarse en la declaración *augment "/ios:native/ios:router"*

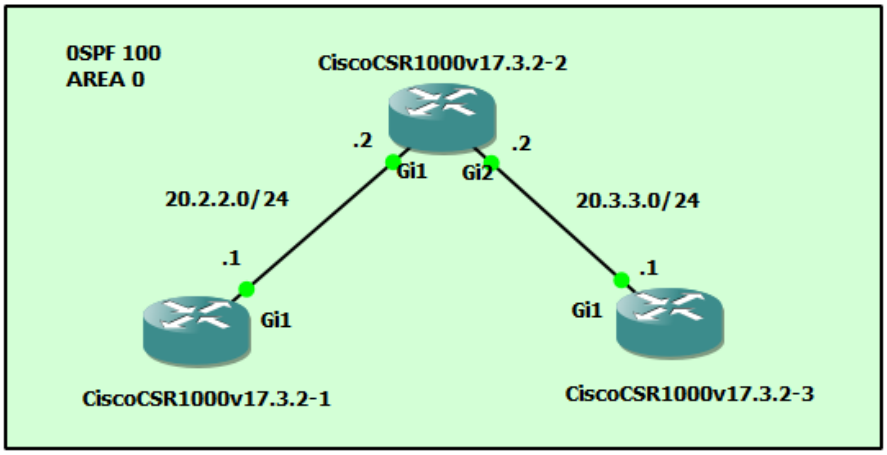


Figura 51. Topología de red para OSPF

El Código 1 fue empleado para configuración de OSPF en la topología de la Figura 51. Se ha decidido dividir el código en cuatro partes para su entendimiento. La primera parte se encarga de crear una conexión NETCONF con autenticación de clave pública, la segunda parte realiza la configuración de OSPF en el almacén de datos de configuración <candidate>. Para ello se creó una variable con el elemento <config>, el cual contiene la jerarquía de datos del modelo de datos Cisco-IOS-XE-ospf. La tercera parte realiza la operación <commit> para establecer la configuración del almacén de datos <candidate> como la configuración de ejecución del dispositivo. La última parte guarda la configuración de ejecución en la configuración de inicio, para ello se empleó la operación <save-config>.

```
1.  ##########################################################
2.  ###Programa para configurar ospf
3.  ##########################################################
4.
5.  from socket import timeout
6.  from ncclient import manager  # Importar clase "manager"
7.  import xml.dom.minidom #Importar el módulo xml.dom.minidom
8.  from ncclient import xml_ # Importar clase "xml_"
9.
10.
11. ##Conexión SSH con clave pública
12.
13. #Administrador de contexto que mantiene "viva" la conexión
14. #ncclient durante la duración del contexto.
15. with manager.connect(
16.     host="192.168.2.1", # Nombre de host o la dirección IP del dispositivo
17.     port="830", # Puerto para la conexión SSH con NETCONF
18.     username="admin2", # Nombre del cliente SSH para la autenticación
19.     #de usuario
20.     hostkey_verify=False, # No verifica las claves de host contenidas
21.     # en ~/.ssh/known_hosts
22.     device_params={"name":"csr"} # Parámetros de los controladores
23.     # de dispositivos
24. ) as m:#Instancia de la clase “manager” (Objeto de sesión NETCONF)
25.     '''
26.     ##Listar capacidades del agente (CSR1K)
```

```
27.
28.     print("Lista de capacidades soportadas en CSR1K:\n")
29.     #Lazo for para para listar las capacidades almacenadas en
30.     #la lista m.server_capabilities
31.     for capability in m.server_capabilities:
32.         print(capability)
33.     '''
34.
35.     ##Metódo edit_config()
36.
37.     #Elemento <config> para configurar ospf
38.     #CISCO-IOX-XE-native.
39.     netconf_ospf = """
40.     <config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
41.       <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
42.         <router>
43.           <router-ospf xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-ospf">
44.             <ospf>
45.               <process-id>
46.                 <id>{ID_proceso}</id>
47.                 <network>
48.                   <ip>{Direc_red}</ip>
49.                   <wildcard>{masc_wildcard}</wildcard>
50.                   <area>{area}</area>
51.                 </network>
52.               </process-id>
53.             </ospf>
54.           </router-ospf>
55.         </router>
56.       </native>
57.     </config>
58.     """
59.     print("Configuración OSPF")
60.     #Se pide al usuario la información para configurar OSPF
61.     ospf_config={}
62.     ospf_config["Proceso_ID"]=input("Ingrese el ID del proceso OSPF: ")
63.     ospf_config["Direccion_red"]=input("Ingrese la dirección de red: ")
64.     ospf_config["Mascara_wildcard"]=input("Ingrese la máscara wildcard: ")
65.     ospf_config["Area"]=input("Ingrese la área OSPF: ")
66.
67.     #Se asigan los elementos del diccionario al elemento <config>
68.     data_ospf=netconf_ospf.format(
69.         ID_proceso=ospf_config["Proceso_ID"],
70.         Direc_red=ospf_config["Direccion_red"],
71.         masc_wildcard=ospf_config["Mascara_wildcard"],
72.         area=ospf_config["Area"]
73.     )
74.
75.     #Operación <edit config> en almacén de datos <candidate> y usando
76.     #elemento <config>
77.     netconf_reply_ospf = m.edit_config(target="candidate",config= data_ospf)
78.     #La variable netconf_reply_ospf almacena la respuesta del
79.     #servidor al finalizar la operación edit_config()
80.
81.     #Imprimir la variable netconf_reply_ospf
82.     print(xml.dom.minidom.parseString(netconf_reply_ospf.xml).toprettyxml())
83.
84.     ##Metódo commit()
85.
86.     #Operación <commit> sin emplear confirmación
87.     netconf_reply = m.commit(confirmed=False)
88.     #La variable netconf_reply almacena la respuesta del
89.     #servidor al finalizar la operación commit()
90.
91.     #Imprimir la variable netconf_reply
```

```
92.     print(xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml())
93.
94.     ##Metódo dispatch()
95.
96.     #Llamamos explícitamente al RPC <save-config> desde
97.     #el modelo de datos Cisco-ia
98.     netconf_save="""
99.     <save-config xmlns="http://cisco.com/yang/cisco-ia"/>
100.        """
101.       ##Se usa el metódo dispatch() para enviar al servidor
102.       #el RPC personalizado <save-config>
103.       netconf_reply_d = m.dispatch(xml_.to_ele(netconf_save))
104.       #La variable netconf_reply_d almacena la respuesta del
105.       #servidor al finalizar la operación dispatch()
106.
107.       #Imprimir la variable netconf_reply_d con formato XML
108.       print(xml.dom.minidom.parseString(netconf_reply_d.xml).toprettyxml())
```

Código 1. Programa para configurar OSPF

Los siguientes pasos muestran el proceso que fue realizado, para la configuración de OSPF. Es importante recalcar que el proceso mostrado a continuación empieza luego de terminar la configuración IP de las interfaces. Dicha configuración no se muestra, debido a que el ejercicio se enfoca solo en la configuración de OSPF.

1. **Configuración de OSFP en el Cisco CSR1K 1**

   El administrador se conectó mediante una sesión NETCONF al Agente y luego empleó el programa del Código 1 para la configuración de OSPF (ver Figura 52). La configuración realizada se observa en la Figura 53 y los resultados de la misma se observan en la Figura 54 y Figura 55.

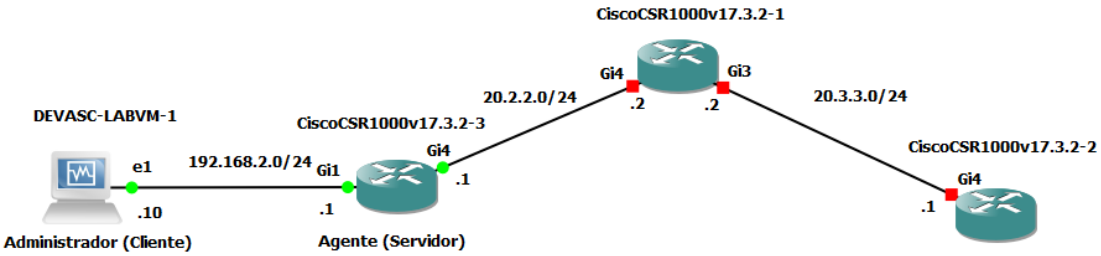


Figura 52. Topología de red para la configuración de OSPF en Cisco CSR1K 1

```
devasc@labvm:~/labs/devnet-src$ /bin/python3 /home/devasc/labs/devnet-src/netconf/ncclient-ospf.py
Configuración OSPF
Ingrese el ID del proceso OSPF: 100
Ingrese la dirección de red: 20.2.2.0
Ingrese la máscara wildcard: 0.0.0.255
Ingrese la área OSPF: 0
```

Figura 53. Configuración OSPF en Cisco CSR1K 1

```
<?xml version="1.0" ?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns
:netconf:base:1.0" message-id="urn:uuid:ee40eaa1-672c-4716-ade8-050120533e5a">
        <ok/>
</rpc-reply>

<?xml version="1.0" ?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns
:netconf:base:1.0" message-id="urn:uuid:55b9754c-ccaa-4bcd-abc9-5ed1866a7e05">
        <ok/>
</rpc-reply>

<?xml version="1.0" ?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns
:netconf:base:1.0" message-id="urn:uuid:c7ad7beb-42bc-4640-8d6d-1e88d584436b">
        <result xmlns="http://cisco.com/yang/cisco-ia">Save running-config successful</resu
lt>


</rpc-reply>
```

Figura 54. Respuestas exitosas al ejecutar el programa en Cisco CSR1K 1

```
router ospf 100
 network 20.2.2.0 0.0.0.255 area 0
!
```

Figura 55. OSPF configurada en Cisco CSR1K 1

2. Configuración de OSFP en el Cisco CSR1K 2

De la misma forma, el administrador se conectó mediante una sesión NETCONF al Agente y luego empleó el programa del Código 1 para la configuración de OSPF (ver Figura 56). Fue necesario emplear el programa dos veces para realizar la configuración, esto se observa en la Figura 57 y Figura 58. Los resultados se observan en la Figura 59 y Figura 60.

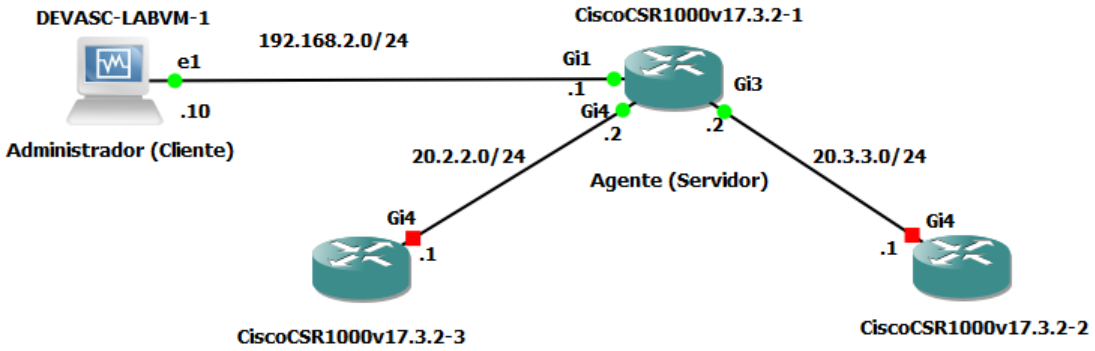


Figura 56. Topología de red para la configuración de OSPF en Cisco CSR1K 2

```
devasc@labvm:~/labs/devnet-src$ /bin/python3 /home/devasc/labs/devnet-src/netconf/ncclient-ospf.py
Configuración OSPF
Ingrese el ID del proceso OSPF: 100
Ingrese la dirección de red: 20.2.2.0
Ingrese la máscara wildcard: 0.0.0.255
Ingrese la área OSPF: 0
```

Figura 57. Configuración OSPF en Cisco CSR1K 2 (Parte 1)

```
devasc@labvm:~/labs/devnet-src$ /bin/python3 /home/devasc/labs/devnet-src/netconf/ncclient-ospf.py
Configuración OSPF
Ingrese el ID del proceso OSPF: 100
Ingrese la dirección de red: 20.3.3.0
Ingrese la máscara wildcard: 0.0.0.255
Ingrese la área OSPF: 0
```

Figura 58 Configuración OSPF en Cisco CSR1K 2 (Parte 2)

```
<?xml version="1.0" ?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns
:netconf:base:1.0" message-id="urn:uuid:f540aa15-0708-4e86-8e15-1e2a8fb1cbe9">
        <ok/>
</rpc-reply>

<?xml version="1.0" ?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns
:netconf:base:1.0" message-id="urn:uuid:fb63ab05-7b48-47b5-be21-8a070f124fc7">
        <ok/>
</rpc-reply>

<?xml version="1.0" ?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns
:netconf:base:1.0" message-id="urn:uuid:68be07d8-c9b1-4703-801d-9bc04c728031">
        <result xmlns="http://cisco.com/yang/cisco-ia">Save running-config successful</resu
lt>


</rpc-reply>
```

Figura 59. Respuestas exitosas al ejecutar dos veces el programa en Cisco CSR1K 2

```
router ospf 100
 network 20.2.2.0 0.0.0.255 area 0
 network 20.3.3.0 0.0.0.255 area 0
!
```

Figura 60. OSPF configurada en Cisco CSR1K 2

3. Configuración de OSFP en el Cisco CSR1K 3

Por ultima vez, el administrador se conectó mediante una sesión NETCONF al Agente y luego empleó el programa del Código 1 para la configuración de OSPF (ver Figura 61). La configuración realizada se observa en la Figura 62 y los resultados de la misma se observan en la Figura 63 y Figura 64.

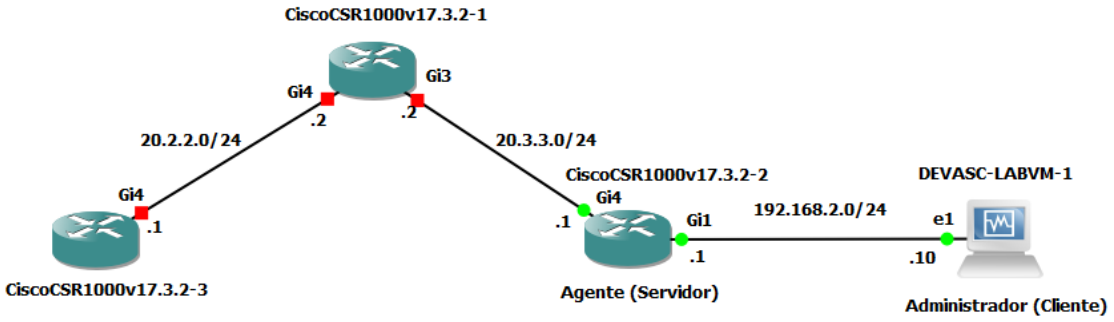


Figura 61. Topología de red para la configuración de OSPF en Cisco CSR1K 3

```
devasc@labvm:~/labs/devnet-src$ /bin/python3 /home/devasc/labs/devnet-src/netconf/ncclient-ospf.py
Configuración OSPF
Ingrese el ID del proceso OSPF: 100
Ingrese la dirección de red: 20.3.3.0
Ingrese la máscara wildcard: 0.0.0.255
Ingrese la área OSPF: 0
```

Figura 62. Configuración OSPF en Cisco CSR1K 3

```
<?xml version="1.0" ?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns
:netconf:base:1.0" message-id="urn:uuid:38d2973b-0e94-476d-877f-0ccb15820861">
        <ok/>
</rpc-reply>

<?xml version="1.0" ?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns
:netconf:base:1.0" message-id="urn:uuid:39f92c4c-1a02-49c3-bd05-09f8e23deebd">
        <ok/>
</rpc-reply>

<?xml version="1.0" ?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns
:netconf:base:1.0" message-id="urn:uuid:b40fb0c2-1636-4d0d-9035-3c1daf72dfdc">
        <result xmlns="http://cisco.com/yang/cisco-ia">Save running-config successful</resu
lt>
```

Figura 63. Respuestas exitosas al ejecutar dos veces el programa en Cisco CSR1K 3

```
router ospf 100
 network 20.3.3.0 0.0.0.255 area 0
!
```

Figura 64. OSPF configurada en Cisco CSR1K 3

4. Pruebas de Funcionamiento

Se apagó la máquina virtual DEVASC y se encendieron los tres routers juntos como se muestra en la Figura 65. Como prueba de funcionamiento se verificaron las tablas de enrutamiento del Cisco CSR1K 1 y Cisco CSR1K 2, demostrando de esta forma la correcta configuración de OSPF. (ver Figura 66 y Figura 67)

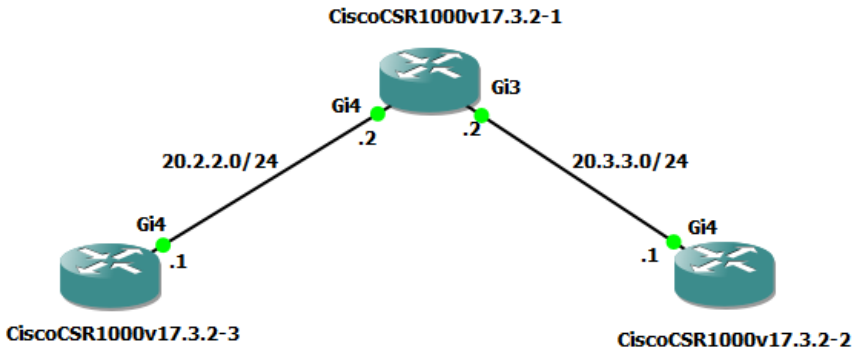


Figura 65. Topología de red para probar funcionamiento de OSPF

```
CSR1k#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, m - OMP
       n - NAT, Ni - NAT inside, No - NAT outside, Nd - NAT DIA
       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
       H - NHRP, G - NHRP registered, g - NHRP registration summary
       o - ODR, P - periodic downloaded static route, l - LISP
       a - application route
       + - replicated route, % - next hop override, p - overrides from PfR
       & - replicated local route overrides by connected

Gateway of last resort is not set

      20.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
O        20.2.2.0/24 [110/2] via 20.3.3.2, 00:05:38, GigabitEthernet4
C        20.3.3.0/24 is directly connected, GigabitEthernet4
L        20.3.3.1/32 is directly connected, GigabitEthernet4
CSR1k#
```

Figura 66. Tabla de enrutamiento en Cisco CSR1K 3

```
CSR1k#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, m - OMP
       n - NAT, Ni - NAT inside, No - NAT outside, Nd - NAT DIA
       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
       H - NHRP, G - NHRP registered, g - NHRP registration summary
       o - ODR, P - periodic downloaded static route, l - LISP
       a - application route
       + - replicated route, % - next hop override, p - overrides from PfR
       & - replicated local route overrides by connected

Gateway of last resort is not set

      20.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
C        20.2.2.0/24 is directly connected, GigabitEthernet4
L        20.2.2.1/32 is directly connected, GigabitEthernet4
O        20.3.3.0/24 [110/2] via 20.2.2.2, 00:06:50, GigabitEthernet4
CSR1k#
```

Figura 67. Tabla de enrutamiento en Cisco CSR1K 1

c. Conclusiones y Recomendaciones.

Dependerá de los estudiantes escribir sus propias conclusiones y recomendaciones.