PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR FACULTAD DE INGENIERÍA

CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

PLAN DE DISERTACIÓN

Tema: SIMULACIÓN Y EVALUACIÓN DE UN SISTEMA DE GESTIÓN DE RED AUTOMATIZADO MEDIANTE LOS PROTOCOLOS NETCONF Y RESTCONF.

AUTOR.

JEREMY FRANCISCO MORENO ESPINOZA

QUITO 06, SEPTIEMBRE DE 2023

RESUMEN

El rápido crecimiento de las redes ha hecho que administrarlas mediante la CLI sea cada vez más complicado. La CLI, al ser un proceso manual, es propensa a errores y consume mucho tiempo. Afortunadamente, hay una solución: adoptar protocolos modernos como NETCONF y RESTCONF, junto con la programabilidad en Python. Estos protocolos permiten automatizar tareas y reducir los tiempos de configuración, mientras que Python facilita la creación de scripts y herramientas para una gestión más eficiente. Utilizando estos recursos, las redes se pueden administrar de manera más precisa y rápida, resultando en configuraciones más sólidas y menos errores.

TABLA CONTENIDO

Comentado [EC1]: Actualizar la tabla completa

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO I: Introducción

1.1 Tema:

Simular un entorno LAN y evaluar la automatización de la gestión de redes mediante los protocolos NETCONF Y RESTCONF.

1.2 Justificación:

En la actualidad, el uso de automatización en las redes ha tenido un impacto significativo debido a que presenta una gran eficiencia operativa, permitiendo realizar la administración de manera correcta. Esto permite a los administradores de red, mediante los protocolos NETCONF y RESTCONF, crear scripts para el cambio de configuraciones e implementar nuevas políticas dentro de la red de forma mucho más rápida y precisa, reduciendo el tiempo dedicado a cada una de estas tareas.

Al adoptar NETCONF y REST dentro de nuestras redes, generamos una mejor interoperabilidad entre dispositivos de red de diferentes marcas, lo que reduce considerablemente el tiempo del personal técnico en el proceso de configuraciones y monitoreo de la red, dejando de lado la dependencia de soluciones propietarias de cada una de las marcas.

A su vez, debido al uso de dichos protocolos en los entornos de red, se permite un mayor acceso a los recursos de red gracias al uso del protocolo HTTP, lo que proporciona una gran flexibilidad y escalabilidad dentro de nuestras implementaciones. Esto nos permite enfrentar el constante crecimiento de la red de forma más estructurada y precisa, llevando un control mucho más detallado de la infraestructura de red.

1.3 Planteamiento del Problema

En principio, la administración de una red resulta más sencilla en cuanto a manejabilidad debido a los pocos dispositivos involucrados. Sin embargo, mientras más crecimiento exista, aumenta su complejidad. El agregar más dispositivos, servicios y usuarios finales resulta en un aumento en la dificultad de su gestión. Esto se convierte en un enfoque de administración complejo en donde resulta muy difícil realizar la configuración, el monitoreo y la seguridad. Al ser las redes configuradas por humanos, estas son propensas a errores y a pasar por alto detalles importantes. Esto puede conllevar a interrupciones en los servicios de red y crear grandes brechas de seguridad. Generando molestias en los usuarios finales y a su vez un uso poco eficiente de los recursos disponibles.

Al no tener una administración automatizada dentro de la red, se pierde la consistencia en las configuraciones, generando así un mal funcionamiento de dicha red y a su vez existen caídas constantes del servicio. De manera que, el tamaño de la infraestructura de red impacta de forma significativa en tareas como la supervisión, que se da de una forma incompleta, pasando por altos configuraciones y detalles técnicos que juegan un papel importante dentro del desempeño de la red. Estos problemas son difíciles de identificar y corregir debido a la complejidad de la red.

1.4 Objetivos

1.4.1 General

Simular un entorno LAN y evaluar la automatización de la gestión de redes mediante los protocolos NETCONF Y RESTCONF.

1.4.2 Específicos

- Desarrollar el marco teórico sobre la automatización en redes y sus herramientas.
- Desarrollar una propuesta de topología en un entorno LAN simulado en herramientas como GSN3.
- Aplicar conceptos y scripts de automatización mediante los protocolos NETCONF y RESTCONF en el entorno LAN.
- Desarrollar una interfaz para la administración de puertos y VLANs.

1.5 Antecedentes

Los administradores de red debido al constante crecimiento que presenta la infraestructura de red en la actualidad se enfrentan con mayor frecuencia a procesos repetitivos que son propensos a fallos debido a la magnitud de las redes en los cuales se pierden una cantidad de tiempo importante es por ello que juegan un papel muy importante las herramientas de automatización debido a la eficiencia en los procesos operativo y la agilidad en aspectos de monitoreo hace que la recuperación ante fallos sea mucho más rápida. Debido a esto se toma como referencia los siguientes trabajos previos como son:

Network Automation using Python (Milios, George, 2021). En este estudio, el autor aborda las herramientas más utilizadas para la automatización en redes y su funcionamiento para lograr automatizar procesos que requieren una configuración manual en cada uno de los dispositivos, siendo estas una tarea que consume demasiado tiempo, como los procesos de generar Backus y restaurar las configuraciones, hasta la generación de nuevas políticas de seguridad dentro del firewall. Se hace bastante énfasis en la configuración de equipos de distintas marcas desde una aplicación web.

Automatización de redes informáticas con Python(Conde Gil, 2021). desarrolla principalmente el uso de las librerías de Python para la automatización en redes y nos presenta como el uso de la CLI para la configuración de redes resulta un proceso demasiado tedioso y tardado el cual resulta demasiado propenso a errores dentro de la red e inviable para la resolución de problemas cuando la infraestructura de red es demasiado grande.

Finalmente, Desarrollo de un servicio API REST para simplificar la administración de redes multi-vendor y multi-entorno(Torrejón Vázquez, 2024). Nos presenta la necesidad de desarrollar un entorno que pueda ser fácilmente adaptado a entornos híbridos de nube e infraestructura tradicional para poder ajustarnos a estos cambios. El administrador de red debe adoptar la figura de NetDevOps para lograr agilizar las tareas y simplificar la administración de estos entornos. En el estudio, se propone una API REST en Python en la web, que permita gestionar los dispositivos de red independientemente de si se encuentran en un entorno de nube o en un entorno tradicional, y del proveedor.

1.6 Alcance

El proyecto tiene como objetivo simular una topología de tres capas de baja complejidad utilizando el software GNS3. Se automatizará la administración de las VLANs. Para ello, se utilizarán imágenes de Junos OS para simular los switches de la red, mientras que para los usuarios finales se emplearán las VPCs. Los protocolos NETCONF y RESTCONF serán responsables de la automatización de la red.

Toda la administración de la automatización se llevará a cabo desde una sola máquina virtual. Debido a las limitaciones del equipo, no se podrán agregar muchos elementos a la simulación para evitar problemas de funcionamiento.

1.7 Metodología

Se llevará a cabo un análisis cuantitativo con el objetivo de examinar los antecedentes para abordar un caso particular. Se procederá desarrollando una propuesta de topología en software y herramientas de simulación. Una vez que la red esté definida y todos sus elementos estén en su lugar, se aplicarán los protocolos NETCONF y RESTCONF para observar el funcionamiento de la administración de VLANs de manera automatizada.

CAPÍTULO II: Fundamentación Teórica

2.1 Gestión de Redes Tradicional

La gestión de la red es el proceso de configurar, supervisar y mantener una red confiable, lo que garantiza la conectividad entre los dispositivos y las personas o las aplicaciones de software que los utilizan. Se puede describir la gestión de la red de muchas maneras diferentes, pero generalmente incluye las prácticas de preparación, configuración, seguridad y medición que los equipos de operaciones de red usan para desarrollar y mantener la eficiencia en la infraestructura de red de su empresa (Red Hat, 2019). Este conjunto de prácticas se ha llevado de manera manual y descentralizada lo que implica una cantidad de tiempo demasiado alta. Esto hace que estos procesos sean muy complejos y más cuando se tiene una infraestructura sumamente grande.

Dentro de los aspectos que resultan claves en una gestión de red tradicional tenemos los siguientes:

- **Configuración Red:**

Uno de los procesos que conlleva más tiempo a los administradores de red es la configuración de los equipos mediante la CLI. Esto implica que para poder configurar cada dispositivo se debe acceder de forma individual por lo que resulta poco práctico cuando tenemos una infraestructura muy grande y a su vez el control de dichas configuraciones se ve afectado teniendo como resultado poca consistencia dentro de las configuraciones.

- **Operación de Red:**

El objetivo principal de las operaciones de red es asegurarse de que la red funcione sin problemas y de manera eficiente. Esto significa estar atentos a lo que sucede en la red todo el tiempo y resolver cualquier problema que surja de manera rápida y efectiva.

En un entorno de gestión de red tradicional, los administradores tienen la tarea de vigilar lo que sucede en la red utilizando herramientas básicas y observando directamente los dispositivos. A menudo, cuando algo no funciona bien, lo descubren debido a que los usuarios lo hacen saber ya que no se cuenta con herramientas avanzadas para detectar problemas automáticamente y un sistema de alertas.

2.2 Automatización de Redes

La automatización de la red es la combinación de software y procesos para aprovisionar, configurar y administrar todos los dispositivos físicos y virtuales dentro de la red de una organización. Con las funciones diarias automatizadas y los procesos repetitivos optimizados y controlados, la disponibilidad del servicio de red y la experiencia general del usuario mejoran como resultado(Fortinet, n.d).

Esta automatización es usada en diversas tareas dentro de nuestra red. Dentro de sus principales ventajas es la mejora de la eficiencia, la reducción de los costos y reduce los errores humanos dentro de las configuraciones de nuestra red. Esto permite tener una mejor administración de la red teniendo como objetivo mejorar la eficiencia y la confiabilidad de la red.

2.2.1 Beneficios de una red automatizada

- **Reducción de riesgos**

La reducción de errores humanos no solo disminuye el riesgo de fallos en la red, sino también las vulnerabilidades de seguridad y posibles violaciones de seguridad. Al automatizar nuestras redes, estas tecnologías permiten aplicar parches de seguridad y actualizaciones con mayor facilidad, asegurando que nuestros sistemas estén siempre actualizados con los últimos parches. Esto reduce la necesidad de que el personal actualice manualmente los equipos, lo cual podría llevar mucho tiempo debido a la gran cantidad de equipos en nuestra infraestructura. Además, verificar que estos parches o actualizaciones no afecten la red puede realizarse de manera más efectiva.

- **Cambios más rápidos**

Uno de los mayores desafíos en la gestión de redes es realizar cambios en nuestros equipos, ya que pueden surgir inconvenientes que creen brechas de seguridad en la red. Mediante la automatización de las redes, se pueden realizar ajustes rápidos y mantener la consistencia en las configuraciones, lo que facilita la implementación de nuevos cambios y reduce considerablemente los fallos en la red.

- **Redes más confiables**

La automatización permite a la red prevenir y prever futuros problemas. La captura de datos dentro de la red se vuelve mucho más fácil, ya que la recolección de logs de cada dispositivo permite analizar estos datos.

- **Simplificación de la administración**

Esta tecnología permite una gestión más frecuente y sencilla, garantizando un funcionamiento óptimo de la red sin tiempos de inactividad debido al monitoreo constante.

Debido a esto, las organizaciones cada vez hacen más esfuerzos para automatizar la gestión de sus redes debido a los beneficios que representa y al alto ahorro en costes. Las configuraciones ya no son un problema, ya que se pueden aplicar de manera más fácil y coherente en nuestra red con mucho menos esfuerzo que la gestión manual, lo que facilita la escalabilidad de la red.

2.2.2 SDN

Las redes definidas por software (SDN) son una categoría de tecnologías que permiten gestionar una red mediante software. La tecnología SDN permite que los administradores de TI configuren sus redes mediante una aplicación de software. El software SDN es interoperable, lo que significa que debería poder funcionar con cualquier enrutador o conmutador, independientemente del proveedor que lo haya fabricado(Cloudflare, n.d.).

La SDN es posible gracias a la separación del plano de control del plano de datos. Dentro del plano de control se agrupan los procesos que gestionan el tráfico que fluye dentro de la red, mientras que el plano de datos hace referencia a los datos que circulan por la red. Un ejemplo más práctico de lo que realiza el plano de control es el establecimiento de rutas dentro de la red y qué protocolos se deben utilizar. El plano de control establece las rutas de la red y dicta qué protocolos deben utilizarse.

En una red tradicional, cada enrutador y switch debe ser configurado de forma individual debido a que dependen del hardware. Con una red SDN, al separar el plano de control del plano de datos y del hardware físico, la configuración se puede dar de manera centralizada.

2.2.2.1 Beneficios de una SDN

- **Infraestructura de Red Flexible.**

Con una SDN, los administradores de la red pueden configurar servicios y asignar los distintos recursos virtuales, lo cual permite modificar la infraestructura en tiempo real a través del plano de control centralizado. Esto permite optimizar los flujos de datos que se transportan a través de la red, priorizando aplicaciones que requieran mayor demanda.

- **Red más Operativa.**

La gestión de la red se centraliza en el plano de control, lo que hace que la supervisión y el control de nuestra red sean mucho más fáciles al tener un único punto de gestión. Esto facilita la implementación de nuevas políticas dentro de nuestra red. Esta rápida adaptación a los cambios nos permite ajustar nuestros anchos de banda de manera rápida y eficiente.

- **Seguridad Robusta.**

La red SDN nos permite tener mayor visibilidad dentro de toda la red, lo que hace que las amenazas de seguridad sean mucho más visibles. Debido a que se centraliza el plano de control, podremos lograr un mejor control dentro de las políticas de seguridad, lo cual permite distribuir distintas clases de políticas a través de nuestra red con el menor esfuerzo. Esto permite a las redes adaptarse rápidamente ante las nuevas amenazas y vulnerabilidades sin la necesidad de volver a configurar nuestros dispositivos.

2.3 Diferencias entre SDN e Infraestructura Tradicional

La mayor diferencia entre SDN y las redes tradicionales radica en su infraestructura: SDN se basa en software, mientras que las redes tradicionales se basan en hardware. Esto significa que, en SDN, el control de la red se realiza mediante software, lo

que le otorga una flexibilidad mucho mayor en comparación con las redes tradicionales. Con SDN, los administradores pueden gestionar la red, realizar cambios en la configuración, asignar recursos y expandir la capacidad de la red, todo desde una única interfaz de usuario centralizada, sin la necesidad de adquirir más hardware. Es como tener la capacidad de ajustar las luces de una habitación desde un solo interruptor en lugar de tener que ir a cada bombilla individualmente para hacer cambios.

2.4 NETCONF

Tradicionalmente, la gestión de los dispositivos de red se llevaba a cabo mediante una interfaz de línea de comandos, tanto para visualizar las distintas configuraciones o datos informativos de cada equipo. Esto supone un inconveniente, ya que las distintas marcas tienen sus propios comandos de configuración, lo que dificulta su interpretación en redes con dispositivos heterogéneos.

Dentro de la gestión de redes, el protocolo SNMP es comúnmente usado para el intercambio de información de varios dispositivos de red, lo cual resulta bastante útil. Sin embargo, presenta varias limitaciones, como que no distingue entre comandos de configuración y comandos de visualización.

Debido a esto, surge la necesidad de adoptar una forma más organizada y basada en estándares para poder configurar los dispositivos de red, reemplazando los procesos de configuración manual.

2.4.1 Descripción General NETCONF

NETCONF proporciona un mecanismo para instalar, manipular y eliminar la configuración de dispositivos de red. Utiliza una codificación de datos basada en Extensible Markup Language (XML) tanto para los datos de configuración como para los

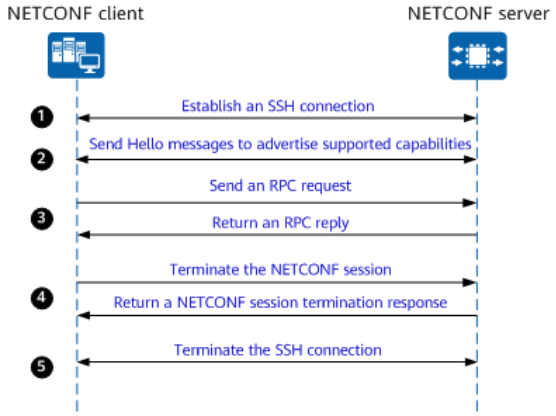
mensajes del protocolo. NETCONF utiliza un mecanismo basado en llamadas de procedimiento remoto (RPC) para facilitar la comunicación entre un cliente y un servidor. El cliente puede ser un script o una aplicación que se ejecuta como parte de un gestor de red. El servidor suele ser un dispositivo de red (switch o enrutador). Utiliza Secure Shell (SSH) como capa de transporte a través de los dispositivos de red (Xiao et al., 2010).

2.4.2 Requisitos del Protocolo de Transporte

NETCONF puede ser implementar sobre cualquier protocolo de transporte que brinde un conjunto de requisitos para asegurar la funcionalidad del protocolo. Mediante estos protocolos de comunicación como SSH y TLS cada par de la comunicación asegura que tiene un método de autentificación y de cifrado que permite que la comunicación sea segura entre el cliente y el servidor NETCONF.

- Operación orientada a la conexión
- Autentificación, integridad y confidencialidad
- Consideración XML
- Espacio de nombres

Figura 1
Establecimiento de una sesión NETCONF.



Nota. Proceso del funcionamiento de una sesión NETCOF. Tomado de (Chunrong, 2022).

2.4.3 Modelo RPC

NETCONF se basa en el modelo de comunicación RPC. Utiliza tanto el elemento <rpc> para las solicitudes como el <rpc-reply> para las respuestas, proporcionando así un marco al protocolo para contener tanto las solicitudes como las respuestas.

2.4.3.1 Elementos RPC

Los elementos RPC agrupan las solicitudes NETCONF que son enviadas desde el cliente al servidor. Uno de los componentes principales y obligatorios es el "message-id", el cual identifica de manera única el mensaje. Al recibir una respuesta, este "message-id" se coloca en el elemento <rpc-reply>.

- **Elemento <rpc-reply>:** Este mensaje se envía como respuesta al mensaje <rpc>. Debe contener obligatoriamente un elemento "message-id" que debe ser el mismo que el del mensaje <rpc>. Los elementos de respuesta se codifican como elementos internos del <rpc-reply>.
- **Elemento <rpc-error>:** Este mensaje se envía dentro del mensaje <rpc-reply> si ocurre algún error durante el proceso de la solicitud <rpc>. Si el servidor encuentra varios errores durante el proceso de una solicitud <rpc>, la respuesta <rpc-reply> contendrá varios `<rpc-error>`, aunque el servidor no necesita devolver más de un error.
- **Elemento <ok>:** Este elemento se envía en el mensaje <rpc-reply> si no se produce ningún problema o advertencia durante el procedimiento de una solicitud <rpc>, y se devuelven los datos de la operación realizada.

2.4.4 Operaciones de NETCONF

El protocolo NETCONF nos provee varias operaciones para la administración de la configuración de los dispositivos y poder verificar los estados de configuraciones actuales.

A continuación, se presentan las operaciones y una breve descripción de lo que desempeñan:

- get-config: Recupera información completa o parcial del datastore de cada dispositivo de red.
- edit-config: Esta operación permite cargar toda o partes específicas de la configuración del datastore. En caso de que la configuración en el datastore no exista, esta será creada.

- copy-config: Permite crear o reemplazar las configuraciones completas del datastore con las de otro datastore. Si la configuración del datastore existe, la sobrescribe; de lo contrario, la crea.
- delete-config: Elimina la configuración del datastore. La configuración en ejecución no puede ser eliminada.
- lock: Permite al cliente bloquear la configuración del datastore del sistema del dispositivo. Esto permite a los clientes realizar cambios sin el problema de que otro cliente realice cambios en la misma configuración.
- get: Recupera la configuración en ejecución e información del estado actual del dispositivo.
- close-session: Solicita la terminación de una sesión NETCONF.
- kill-session: Forza la terminación de una sesión NETCONF.

2.5 Modelo YANG

YANG es un lenguaje de modelado de datos. El modelo YANG "Yet Another Next Generation", define una estructura de datos jerárquica que puede utilizarse para operaciones basadas en protocolos de gestión de configuración de red (como NETCONF/RESTCONF). Las operaciones incluyen configuración, datos de estado, llamadas a procedimientos remotos (RPC) y notificaciones. En comparación con el modelo MIB de SNMP, YANG es más jerárquico, puede distinguir entre configuraciones y estados y ofrece una gran extensibilidad(¿Qué Es El Modelo YANG?, 2024).

Con su enfoque jerárquico y orientado a objetos, YANG facilita la representación clara y concisa de la información necesaria para configurar y administrar diferentes dispositivos. Además de ofrecer una gama completa de tipos de datos básicos, YANG

también permite la creación de tipos personalizados para adaptarse a necesidades específicas.

A continuación, podemos observar un ejemplo de un formato YANG para poder visualizar la estructura en formato YANG:

Figura 2
Estructura de formato YANG.

```
module configuracion-interfaz {
  yang-version 1.1;
  namespace "http://ejemplo.com/configuracion-interfaz";
  prefix intf;
  // Declaración de la entidad interfaz
  container interfaz {
    description "Configuración de la interfaz de red";
    leaf nombre {
      type string;
      description "Nombre de la interfaz";
    }
    leaf direccion-ip {
      type inet:ipv4-address;
      description "Dirección IP";
    }
    leaf mascara-subred {
      type inet:ipv4-prefix;
      description "Máscara de subred";
    }
    leaf puerto {
      type inet:port-number;
      description "Número de puerto";
    }
  }
}
```

En este ejemplo, hemos definido un módulo YANG llamado configuración-interfaz que contiene una estructura de datos llamada interfaz. La estructura de datos incluye tres elementos: nombre, dirección IP, máscara de subred y puerto, cada uno con su tipo de dato correspondiente y una descripción que indica su propósito.

2.6 RESTCONF

RESTCONF usa los métodos HTTP para realizar las operaciones dentro de los datastore que están representada por el modelo de datos YANG. Estas operaciones permiten alterar la configuración que se está ejecutando dentro del servidor desde el cliente RESTCONF. Mediante la combinación de HTTP nos brinda la ventaja de desarrollar una API que permite la administración de nuestros equipos de red.

2.6.1 Recursos de RESTCONF

Un recurso en el protocolo RESTCONF puede ser considerado como una colección de datos o un conjunto de métodos permitidos para la manipulación de los datos. Cada uno de estos recursos tiene una representación asociada con un identificador de tipo de dato representado por un ¨Content-Type¨ en el encabezado HTTP.

2.6.2 Métodos de RESTCONF

The RESTCONF protocol uses HTTP methods to identify the CRUD operations requested for a particular resource. The following table shows how the RESTCONF operations relate to NETCONF protocol operations.

- OPTIONS: Este método enviado desde el cliente permite descubrir cuales métodos están soportados por el servidor.
- HEAD: Retorna solo los campos del encabezado esta respuesta es similar al método GET, pero en este caso sin el cuerpo del mensaje.
- GET: Este método permite retornar todos los datos y los metadatos de cada recurso. Este tipo de petición no soporta los recursos de operación. Él servidor solo retornara los datos donde el usuario no tenga los respectivos permisos.

- POST: Este método sirve para crear un recurso o para realizar la llamada de dicho recurso de operación (Operaciones RPC).
- PUT: Este método permite al cliente crear o remplazar un recurso.
- PATCH: Este método permite modificar parciales a un recurso.
- DELETE: Este método permite eliminar un recurso del datastore.

2.7 Librerías Python

Python es un lenguaje ideal para la automatización de redes debido a su legibilidad, facilidad de aprendizaje y amplio soporte de bibliotecas. Es un lenguaje de alto nivel, lo que significa que abstrae muchas complejidades del sistema, permitiendo a los ingenieros de redes concentrarse en sus tareas en lugar de en los matices del lenguaje de programación (Etiris Magazine, 2023).

2.7.1 Requests

La librería requests en Python es una poderosa herramienta que facilita la interacción con servidores web y APIs mediante el envío de solicitudes HTTP. Con requests, puedes realizar una amplia gama de operaciones, desde simples solicitudes GET hasta solicitudes más complejas que involucran métodos como POST, PUT, DELETE, entre otros.

Es posible agregar parámetros de consultas, información de formularios, cargar JSON y enviar encabezados personalizados en tus peticiones. Esto facilita enviar datos extra con tus peticiones, lo cual es esencial para comunicarte con APIs y servicios web que necesitan ciertos datos para operar adecuadamente.

Comentado [CMEP2]: Está con otro tipo de letra

CAPÍTULO III: Diseño de Topología

Comentado [CF3]: Dividir en dos capitulos

3.1 Requerimientos

Para la simulación de este proyecto se utilizará la herramienta de GNS3 para la simulación del entorno LAN y el hipervisor VMWARE para la virtualización de todos los equipos de red. El equipo donde se ejecutará toda la simulación posee especificaciones limitadas por lo cual no fue posible extender en su totalidad a la red. Se detalla a continuación las especificaciones del equipo utilizado:

- Procesador i5-11400H 2.70GHz
- Memoria RAM 40GB DDR4
- Disco Duro 1TB

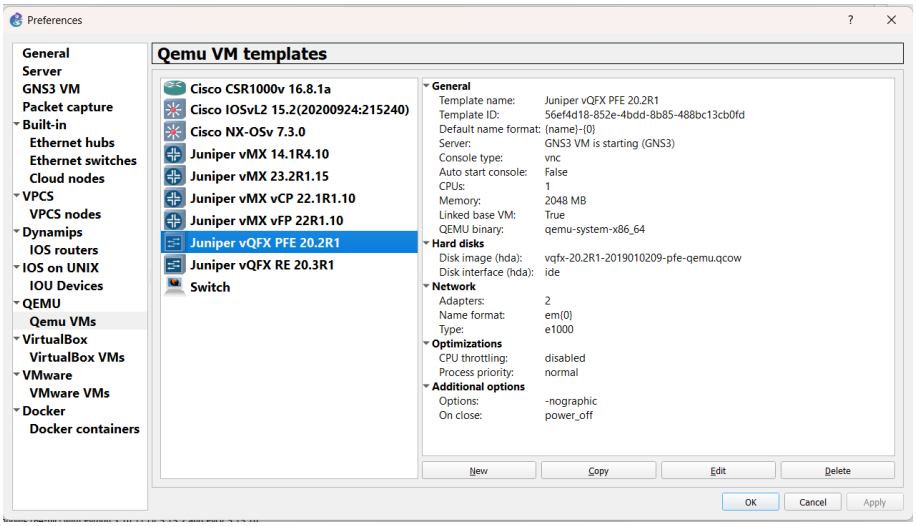
Dentro de la topología se han utilizado dispositivos de red con el sistema operativo juno OS propietario de Juniper Netwoks los cuales nos presentan la compatibilidad requerida con los protocolos NETCONF y RESTCON. En este caso al ser una red LAN se utilizaron switches de la serie QFX los cuales se dividen en dos componentes como son el Routing Engine y el Packet Forward Engine para mejorar su rendimiento.

3.1.1 Packet Forwarding Engine

El Packet Forwarding Engine utiliza circuitos integrados específicos (ASICs) para realizar el switching de paquetes de Capa 2 y Capa 3, búsquedas de rutas y reenvío de paquetes. El Packet Forwarding Engine reenvía paquetes entre las interfaces de entrada y salida.(Juniper Networks, 2023)

Figura 3

Características de la imagen del dispositivo de red vQFX-PFE.

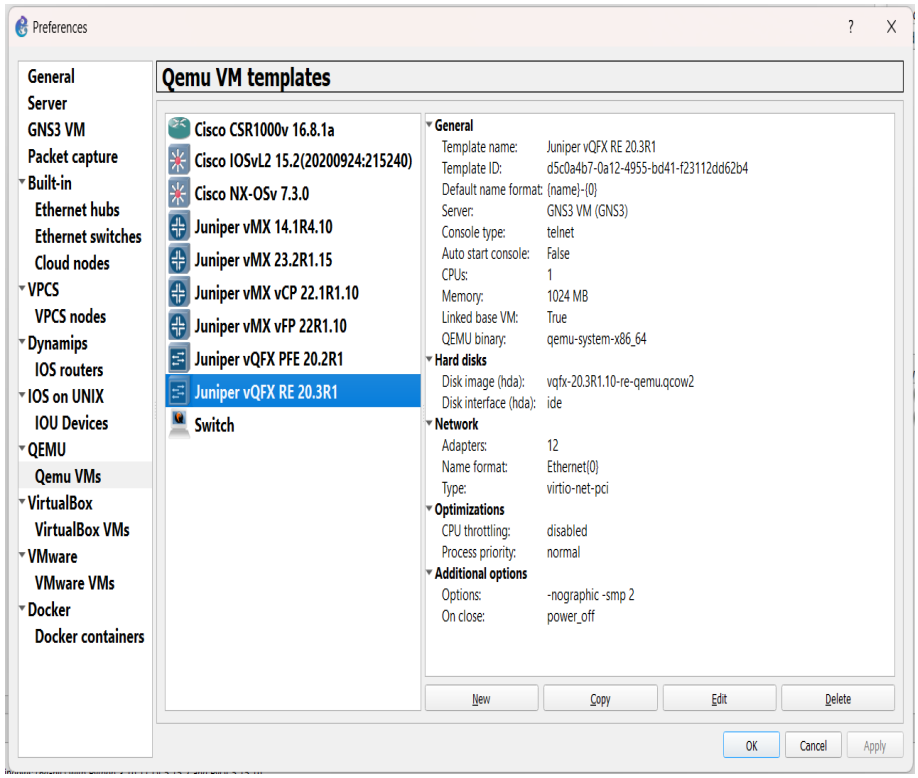


3.1.2 Routing Engine

El Routing Engine consiste en procesos de software de protocolo de enrutamiento que se ejecutan dentro de un entorno de memoria protegida en una plataforma informática de propósito general. El Routing Engine maneja todos los procesos de protocolo de enrutamiento y otros procesos de software que controlan las interfaces de los enrutadores, algunos de los componentes del chasis, la gestión del sistema y el acceso del usuario al enrutador. Estos enrutadores y procesos de software se ejecutan sobre un kernel que interactúa con el Packet Forwarding Engine.(Juniper Networks, 2023).

Figura 4

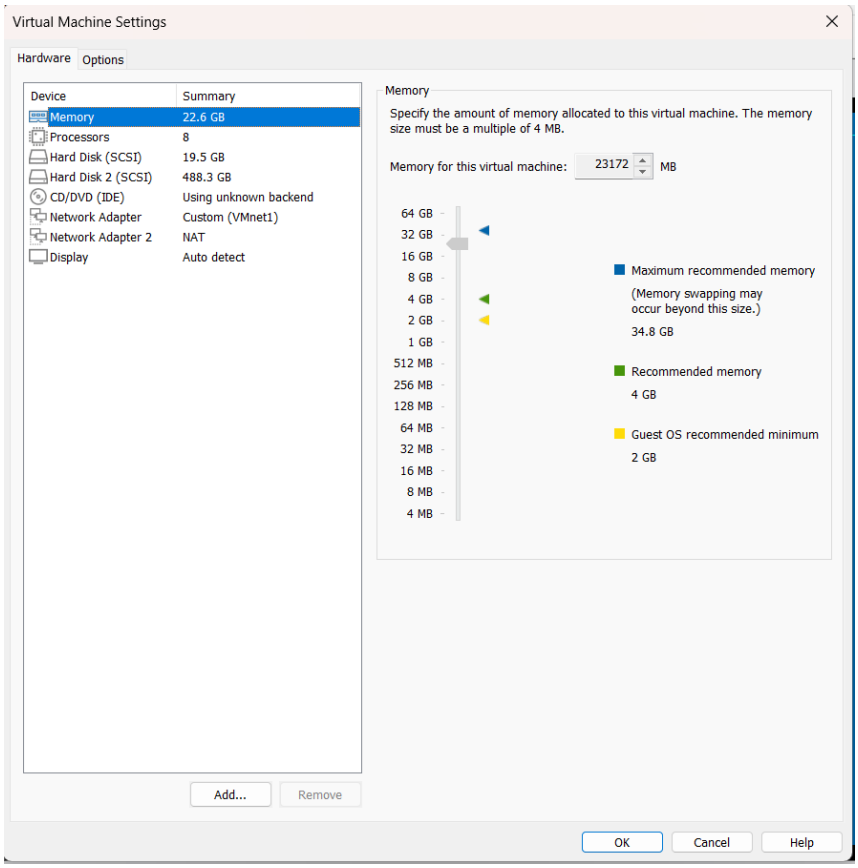
Características de la imagen del dispositivo de red vQFX-RE.



Como parte de las recomendaciones se ha usado VMWARE como hipervisor debido al alto rendimiento frente a otros hipervisores. La máquina virtual de GNS3 cumple la función de ejecutar nuestros dispositivos de red que en este caso son virtualizados con el hipervisor QEMU para lo cual se debe asignar los recursos suficientes para que la máquina virtual se desempeñe sin ningún problema se detallan a continuación os recursos:

Figura 5

Características de la máquina virtual de GNS3.



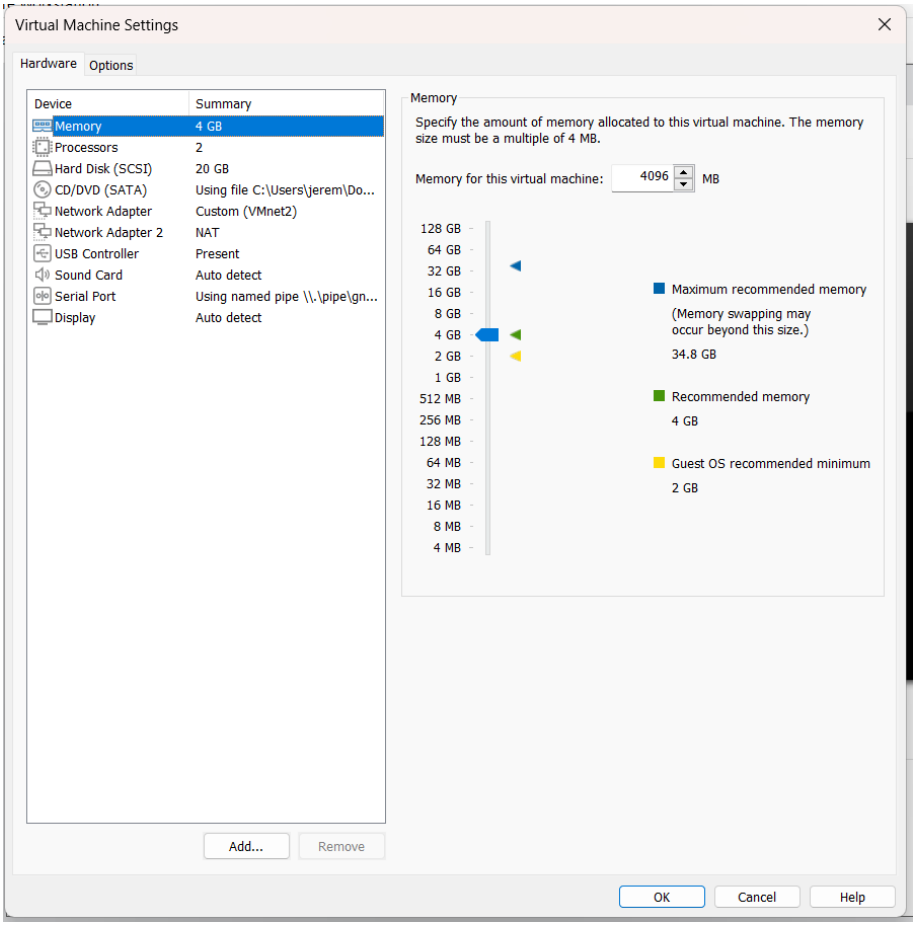
A su vez se cuenta con una máquina virtual Ubuntu que contendrá las diferentes aplicaciones y librerías que permitirá realizar un pequeño sistema que ayudará a gestionar el entorno LAN de mejor manera. Por ello se necesita asignar los recursos suficientes a la máquina virtual para que se desempeñe de forma correcta a continuación se mostraran los recursos asignados:

Comentado [CMEP4]: Cambiar la forma de redacción, no utilizar "conta**mos**", mejor utilizar "se cuenta"

Comentado [CMEP5]: Quitar

Figura 6

Características de la máquina virtual de administración.



3.2 Topología

La topología de red es la forma en la que se distribuyen los distintos dispositivos y sus conexiones. Se diseñó una red jerárquica de tres capas para este entorno de red local (LAN) con un switch de núcleo y un switch de distribución. No fue posible configurar redundancia en estas capas debido a las limitaciones de los dispositivos de red y del equipo donde se realiza la simulación. Además, la infraestructura se completó

con cuatro switches de acceso. A pesar de las limitaciones tecnológicas mencionadas, se implementó esta configuración para maximizar el rendimiento y la eficiencia de la red. Dicha red se presenta en la figura **Figura 7.**

Para simular los usuarios finales se optó por la utilización de VPCS (Virtual PC Simulator) las cuales no representan un consumo tan alto en recurso ya que realizan funcionalidades básicas para probar la comunicación dentro de la red.

Figura 7

Topología del entorno LAN.

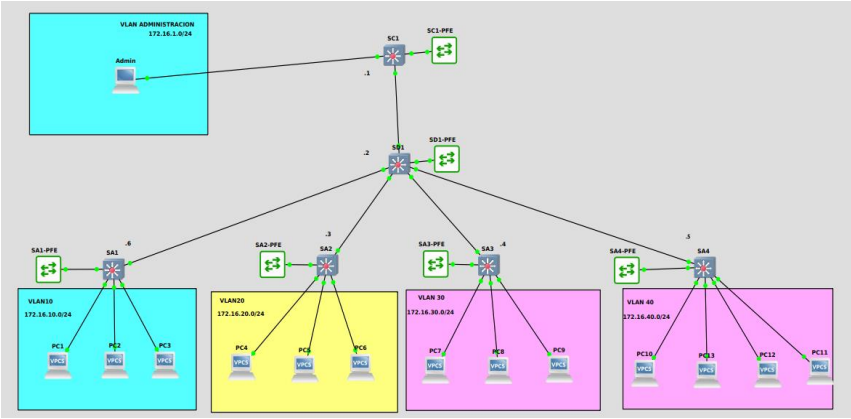
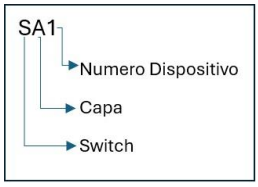


Figura 8

La nomenclatura que se sigue para nombrar los equipos en la topología de red.



Para el correcto funcionamiento de los switch vQFX se debe conectar la interfaz em1 del componente RE a la interfaz em1 del componente PFE de esta forma los dos componentes funcionaran como uno solo mediante la tecnología virtual chassis de Juniper.

3.2.1 Configuración de red

3.2.1.1 Capa de Núcleo

Comentado [CMEP6]: Normalmente, para documentos formales solo se numeran los títulos hasta el tercer nivel

En esta capa, se utilizan dispositivos para conectar redes geográficamente separadas y agilizar la transferencia de datos en la red. Para este caso en concreto, el dispositivo de red se configuró como servidor DHCP para la asignación de las direcciones IP de cada VLAN. Simultáneamente, los switches que funcionan en la capa central realizan la conmutación de paquetes de manera rápida y eficiente. Además, al ser un dispositivo encargado del enrutamiento de campus, se configuró el protocolo OSPF.

3.2.1.2 Capa de distribución

En una red de área local (LAN), la capa de distribución es esencial para agregar datos de la capa de acceso a la capa de núcleo. Su principal tarea es establecer límites claros utilizando listas de acceso y otros filtros con el objetivo de establecer e implementar políticas de seguridad y red. Esta capa proporciona un punto de control para las políticas de red y la gestión del tráfico y es esencial para garantizar el enrutamiento adecuado de paquetes entre las diversas subredes y VLANs de la empresa. Además, el DHCP relay se implementa en la capa de capa para facilitar la asignación de direcciones IP a través de diferentes subredes.

Comentado [CMEP7]: Cambiar por “e”

3.2.1.3 Capa de Acceso

La capa de acceso constituye la base del diseño jerárquico de redes. En esta capa se encuentran los switches de acceso, los cuales se conectan directamente a los dispositivos finales, como computadoras, impresoras y servidores. La función principal

de los switches de la capa de acceso es asegurar la entrega de paquetes a estos dispositivos finales.

CAPÍTULO VI: Implementación

Comentado [CF8]: Dividir en dos capitulos

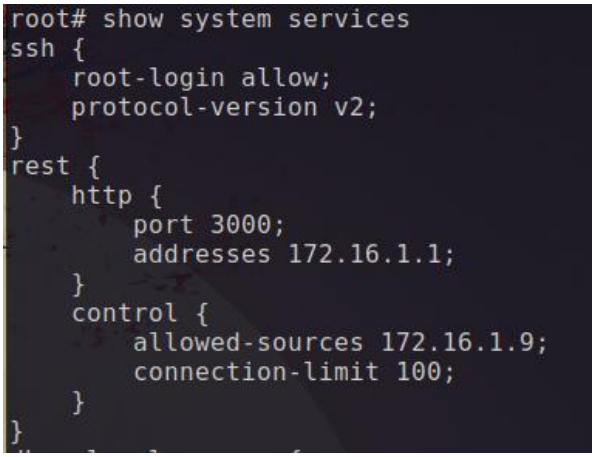
4.1 Desarrollo de scripts en Python

4.1.1 Configuración del Servicio RESTCONF

Este es uno de los pasos más importantes, ya que habilitando el servicio RESTCONF dentro de nuestros dispositivos de red, el cual proporcionará la URL a través de la cual realizar configuraciones mediante peticiones HTTP. En la **Figura 9** se muestra la configuración final de dicho servicio.

Figura 9

Configuración del servicio REST en el switch Core.



4.1.2 Pasos para la Configuración del Servicio RESTCONF

- **Asignación del Puerto**: Definir el puerto en el cual el servicio RESTCONF estará escuchando.
- **Asignación de la Dirección IP**: Especificar la dirección IP en la cual el servicio RESTCONF estará disponible en este caso la dirección está en el segmento de la VLAN administrativa.

- **Configuración de la Dirección IP de Control**: Asignar una dirección IP de control que será la única permitida para acceder al servicio RESTCONF.

Mediante estos pasos logramos que el servicio RESTCONF esté correctamente configurado y accesible solo desde una dirección IP de control, proporcionando seguridad y control en la gestión de las configuraciones de red a través de peticiones HTTP.

4.1.3 Peticiones HTTP

Mediante la biblioteca Requests en Python, realizar todas las peticiones HTTP necesarias. En este caso, en la **Figura 11** se muestra cómo se estructura una petición para obtener la configuración de todo el dispositivo.

Figura 10
Petición POST desde Python para obtener la configuración en ejecución del dispositivo.

La figura muestra un ejemplo sencillo de cómo se crea una solicitud utilizando la biblioteca Requests en Python para obtener toda la configuración de un dispositivo. En este caso concreto, se realiza una petición POST enviando un mensaje RPC en el cuerpo de la solicitud. De este modo, se realizarán las configuraciones correspondientes para la creación y administración de las VLANs.

4.1.4 Configuración de la VLAN

Comentado [CMEP9]: No dejar en esta página, llevarla a la siguiente.

Para este caso en concreto, se pretende desarrollar scripts para realizar un despliegue completo de una VLAN, incluyendo un servicio DHCP que correrá en el switch de núcleo, encargado del direccionamiento IP dentro del entorno simulado. Para ello, se han creado distintas funciones para la configuración de cada componente necesario para el correcto funcionamiento de la VLAN.

A continuación, se destacan los puntos que deben tomarse en cuenta para que el despliegue funcione correctamente:

4.1.4.1 Creación de la VLAN

Esta función se encarga de definir y crear una VLAN y el respectivo id de la misma en los switches mediante la mensajería RPC.

Figura 11
Petición POST desde Python para crear la VLAN.

```
def create_vlan(name: str,id: int ,device:str, interfaces: list):
        FILTER = """<rpc>
        <edit-config>
                <target>
                        <candidate/>
                </target>
                <config>
                        <configuration>
                                <vlans>
                                        <vlan operation="create">
                                        <name>{}</name>
                                        <vlan-id>{}</vlan-id>
                                        </vlan>
                                </vlans>
                        </configuration>
                </config>
        </edit-config>
</rpc>""".format(name, id)
        for key, value in DEVICES.items():
                reply = requests.post("http://{}:3000/rpc".format(value),data=FILTER,
                auth=requests.auth.HTTPBasicAuth(USER, PWD),
                headers={"Accept": "application/json","Content-Type": "application/xml"})
```

En la figura se muestra el mensaje RPC que se enviará a los dispositivos de red para la creación de la VLAN y su identificador único. Como se puede observar, se emplea la operación "create" dentro del objeto "vlan", así especificamos al servidor REST la operación que se desea realizar.

4.1.4.2 Creación Interfaz IRB

Las interfaces IRB (Integrated Routing and Bridging) en los dispositivos Juniper son interfaces lógicas que combinan enrutamiento y puenteado en una sola interfaz. Esto ayuda a que las VLANs no solo se comuniquen entre sí dentro de un switch, sino que también puedan conectarse con otras redes a través del enrutamiento.

Figura 12
Petición POST desde Python para crear la interfaz IRB.

```
def create_irb(id: int):
        for key, value in DEVICES.items():
                aux= value.split(".")
                aux[2] = str(id)
                nuevo = ".".join(aux)
                FILTER = """<rpc>
                <edit-config>
                        <target>
                                <candidate/>
                        </target>
                        <config>
                                <configuration>
                                        <interfaces>
                                                <interface>
                                                        <name>irb</name>
                                                        <unit>
                                                                <name>{}</name>
                                                                <family>
                                                                <inet>
                                                                        <address operation="create">
                                                                        <name>{}/24</name>
                                                                        </address>
                                                                </inet>
                                                                </family>
                                                        </unit>
                                                </interface>
                                        </interfaces>
                                </configuration>
                        </config>
                </edit-config>
        </rpc>""".format(id,nuevo)
                reply = requests.post("http://{}:3000/rpc".format(value),data=FILTER,
                auth=requests.auth.HTTPBasicAuth(USER, PWD),
                headers={"Accept": "application/json","Content-Type": "application/xml"})
```

En la figura se muestra el mensaje RPC que se enviará a los dispositivos de red para la creación de la interfaz IRB y su dirección IP.

4.1.4.3 Configuración de interfaz de acceso

Una interfaz de acceso pertenece a una sola VLAN. Cuando un dispositivo se conecta a una interfaz de acceso, automáticamente se asocia con la VLAN que se ha configurado en ese puerto. Esto permite segmentar la red en diferentes dominios de broadcast.

Figura 13
Petición POST desde Python para asignación de interfaces acceso a cada VLAN.

```
def create_int_acc_vlan(name: str, device: str, interfaces: list):
        for interface in interfaces:
                FILTER = """<rpc>
        <edit-config>
                <target>
                        <candidate/>
                </target>
                <config>
                        <configuration>
                                <interfaces>
                                        <interface>
                                        <name>{}</name>
                                        <unit>
                                                <name>0</name>
                                                <family>
                                                <ethernet-switching>S
                                                        <vlan>
                                                        <members operation="create">{}</members>
                                                        </vlan>
                                                </ethernet-switching>
                                                </family>
                                        </unit>
                                        </interface>
                                </interfaces>
                        </configuration>
                </config>
        </edit-config>
</rpc>""".format(interface,name)
```

En la siguiente figura se observa cómo, mediante la mensajería RPC, se configura la interfaz en modo acceso y se especifica a qué VLAN pertenecerá.

4.1.4.4 Configuración de interfaces troncales

Un enlace troncal en un switch es una conexión que permite transportar el tráfico de varias VLANs entre switches y otros dispositivos de red. Funciona añadiendo etiquetas a las tramas, lo que ayuda a identificar a qué VLAN pertenece cada paquete de datos mientras se envía por el mismo cable físico.

Figura 14

Petición POST desde Python para asignación de interfaces troncales.

```
def create_int_trunk_vlan(name: str):
        interface = "xe-1/0/0"
        devices = ["172.16.1.1","172.16.1.2","172.16.1.3","172.16.1.4","172.16.1.5","172.16.1.6"]
        for key, value in DEVICES.items():
                FILTER = """<rpc>
        <edit-config>
                <target>
                        <candidate/>
                </target>
                <config>
                        <configuration>
                                <interfaces>
                                        <interface>
                                        <name>{}</name>
                                        <unit>
                                                <name>0</name>
                                                <family>
                                                <ethernet-switching>
                                                        <vlan>
                                                        <members operation="create">{}</members>
                                                        </vlan>
                                                </ethernet-switching>
                                                </family>
                                        </unit>
                                        </interface>
                                </interfaces>
                        </configuration>
                </config>
        </edit-config>
</rpc>""".format(interface,name)
                reply = requests.post("http://{}:3000/rpc".format(value),data=FILTER,
                auth=requests.auth.HTTPBasicAuth(USER, PWD),
                headers={"Accept": "application/json","Content-Type": "application/xml"})
```

4.1.4.5 DHCP

DHCP es un servicio de red que asigna direcciones IP y otros parámetros de configuración de manera automática. DHCP elimina la necesidad de asignar direcciones IP manualmente y permite una fácil gestión de la red. Debido a que las direcciones IP se asignan dinámicamente, DHCP puede utilizarse para asignar direcciones IP a un

conjunto de dispositivos temporales, minimizando el uso de direcciones IP y permitiendo la adición de elementos sin intervención.

Figura 15

Petición POST desde Python para creación de DHCP pool.

```
def create_dhcp_pool(name:str, id: int):
        FILTER = """<rpc>
        <edit-config>
                <target>
                        <candidate/>
                </target>
                <config>
                        <configuration>
                                <access>
                                        <address-assignment>
                                                <pool>
                                                        <name>{}</name>
                                                        <family>
                                                        <inet>
                                                                <network>172.16.{}.0/24</network>
                                                                <range>
                                                                <name>{}</name>
                                                                <low>172.16.{}.11</low>
                                                                <high>172.16.{}.250</high>
                                                                </range>
                                                                <dhcp-attributes>
                                                                <router>
                                                                        <name>172.16.{}.1</name>
                                                                </router>
                                                                </dhcp-attributes>
                                                        </inet>
                                                        </family>
                                                </pool>
                                        </address-assignment>
                                </access>
                        </configuration>
                </config>
        </edit-config>
</rpc>""".format(name,id,name,id,id,id)
```

El siguiente mensaje RPC configura el nombre del pool DHCP, el segmento de red de la VLAN, la dirección IP desde la cual se comenzará la asignación del DHCP y la dirección IP del gateway.

4.1.4.6 DHCP Relay

El DHCP Relay es una función que permite a los dispositivos de red reenviar los mensajes DHCP (Protocolo de Configuración Dinámica de Hosts) desde un segmento de red a otro, permitiendo que los clientes DHCP obtengan direcciones IP y otros

parámetros de configuración incluso cuando el servidor DHCP está ubicado en una red diferente.

Figura 16
Petición POST desde Python para creación de DHCP Group.

El grupo de DHCP se crea para cada VLAN especificando la interfaz IRB que estará asignada, de modo que el servidor DHCP pueda asignar las direcciones IP correspondientes.

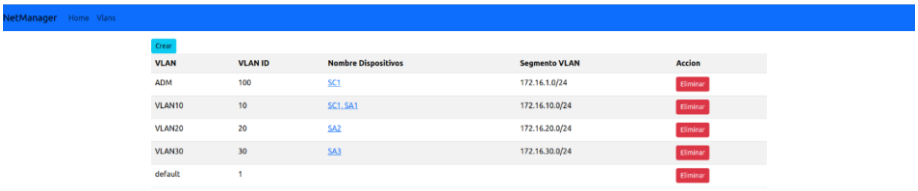
4.2 Desarrollo de interfaz de administración

Flask es un microframework web escrito en Python. Es ligero y flexible, lo que lo hace ideal para el desarrollo de aplicaciones web pequeñas o medianas. Flask complementa el desarrollo de la interfaz de administración proporcionando una base sólida para la creación de aplicaciones web, con enrutamiento simple, integración con Jinja2 para la generación de HTML dinámico, una amplia gama de extensiones y complementos, y una gran flexibilidad y escalabilidad.

4.2.1 Interfaz para visualización de VLANs

Esta interfaz muestra los dispositivos que están conectados a diferentes VLANs, así como información detallada sobre el segmento de red que pertenece a cada VLAN. Esto proporciona una visualización completa de la infraestructura, lo que facilita la identificación de la topología y la asignación de VLANs. Esta integración de datos en la interfaz de administración facilita la planificación de futuras expansiones o ajustes de configuración, lo que garantiza una gestión de red eficiente y precisa.

Figura 17
Interfaz para visualización de VLANs.



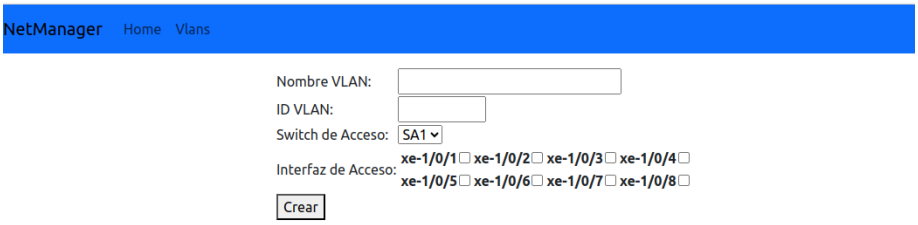
Se puede observar que tenemos el botón "Eliminar", el cual eliminará todas las configuraciones de la VLAN en los dispositivos, haciendo que la configuración quede consistente y reduciendo los errores.

4.2.2 Interfaz para creación VLAN

A través de la interfaz de creación, se pueden establecer todos los parámetros requeridos para crear una VLAN, como el nombre, el ID y el switch al que estará asignada. También, es posible indicar las interfaces de red que estarán asignadas para esa VLAN. Gracias a este proceso automatizado, la creación de la VLAN se vuelve más uniforme y disminuye de manera significativa la probabilidad de errores en las

configuraciones. Esto permite optimizar tanto la eficiencia como la confiabilidad de la red.

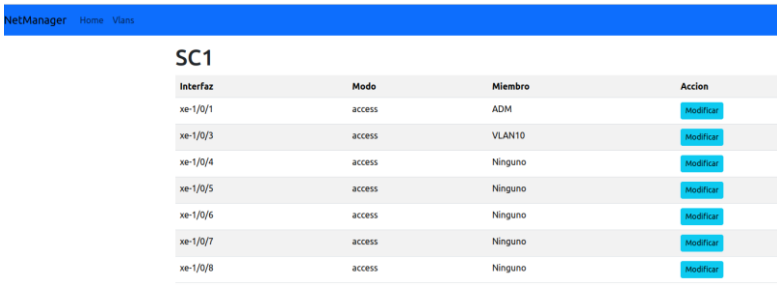
Figura 18
Interfaz para creación de VLANs.



4.2.3 Interfaz de visualización de la asignación de VLANs

La interfaz permite visualizar la asignación de cada una de las VLANs en las interfaces de acceso, logrando así una mejor visualización de los dispositivos conectados. Esto facilita la gestión y el monitoreo de la red, permitiendo identificar rápidamente en qué interfaces se encuentran las distintas VLANs y optimizando la organización y administración de la infraestructura de red.

Figura 19
Interfaz de visualización de la asignación de VLANs a cada interfaz de red.



4.3 Funcionamiento y análisis de resultados

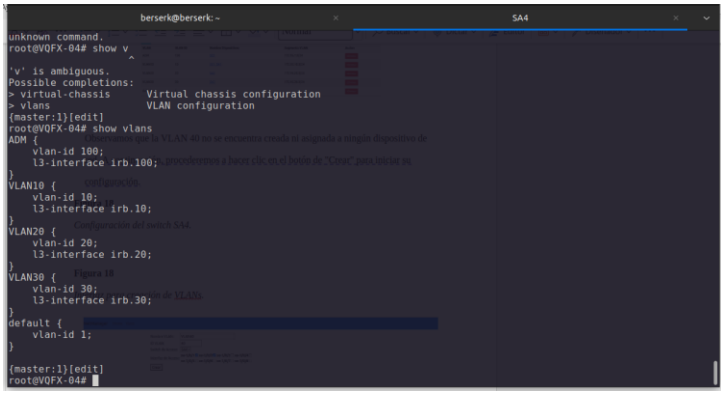
4.3.1 Funcionamiento

En este caso, se probará la creación de la VLAN 40 en el switch SA4, asignando los puertos xe1/0/1 y xe1/0/2.

Figura 20
Interfaz para visualización de VLANs.

La VLAN 40 no se encuentra creada ni asignada a ningún dispositivo de red. A continuación, al hacer clic en el botón de "Crear" iniciara el proceso de configuración.

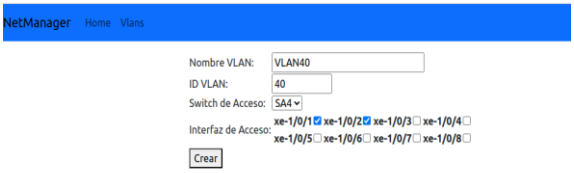
Figura 21
Configuración del switch SA4.



Dentro del dispositivo de red no tenemos creada la VLAN40.

Figura 22

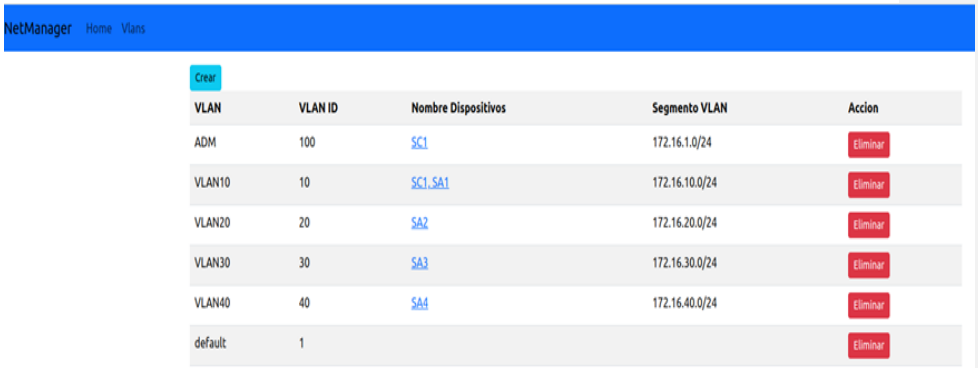
Interfaz para creación de VLANs.



Se ingresan los parámetros correspondientes para la creación de la VLAN, que incluyen el nombre, el ID de la VLAN, el switch de acceso al cual será asignada y las interfaces a las cuales será asignada.

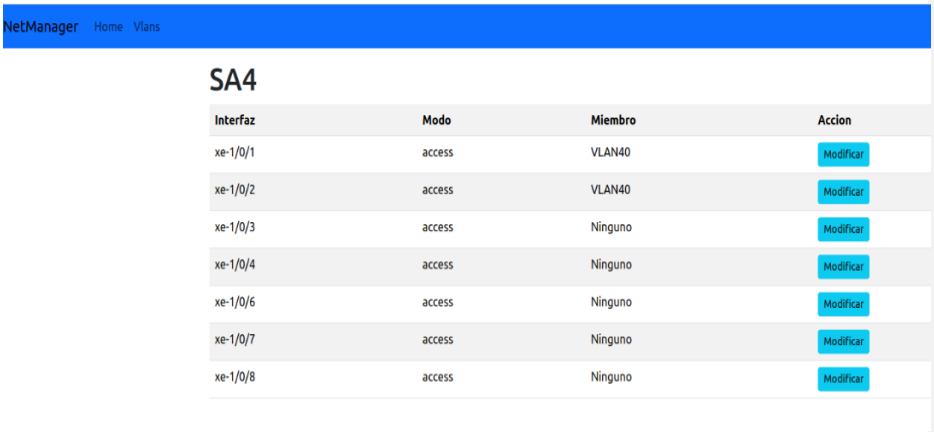
Figura 23

Visualización de VLAN.



Se observa cómo se creó correctamente la VLAN40 en el dispositivo especifico.

Figura 24
Visualización del resumen del dispositivo.

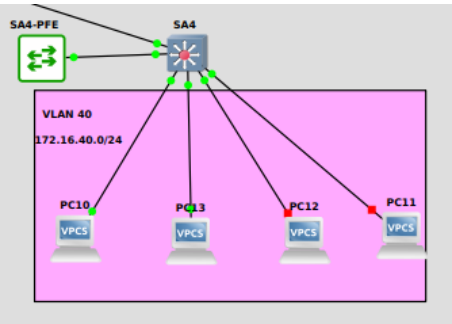


Dentro del resumen del dispositivo la VLAN40 se asignó correctamente dentro de las interfaces de red seleccionadas.

4.3.1.1 Pruebas de conectividad

Se realizará las pruebas para garantizar que la VLAN 40 se haya creado y configurado de manera adecuada. Se verificará si hay conectividad en los hosts asignados a las interfaces xe1/0/1 y xe1/0/2 del switch SA4. También se comprobará la correcta asignación de direcciones IP a estos hosts. Se garantizará que la VLAN 40 funcione correctamente y que los dispositivos conectados logren una comunicación efectiva entre sí dentro de la red segmentada gracias a este proceso.

Figura 25
PC10 conectado a la interfaz xe1/0/1.



Se realizarán pruebas de conectividad hacia el gateway, que está asignado en el SC1, desde los hosts conectados a las interfaces xe1/0/1 y xe1/0/2 del switch SA4. Además, se verificará el direccionamiento DHCP en la PC10 y PC13 para verificar que esté correcto funcionamiento de la VLAN 40.

Figura 26
Prueba PC10 DHCP.

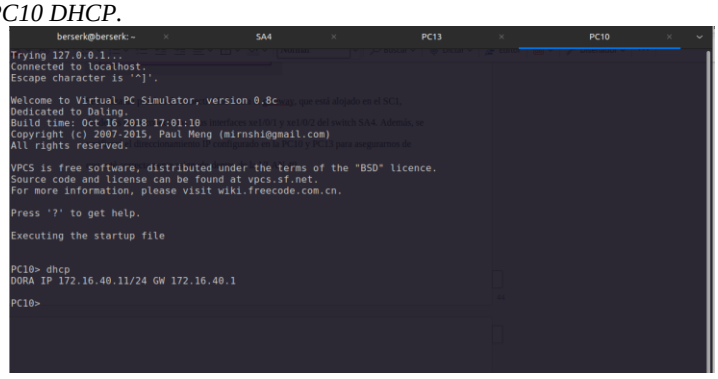
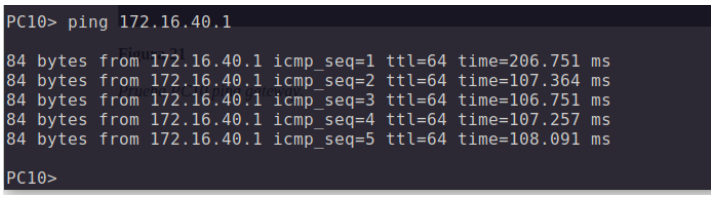


Figura 27
Prueba PC10 ping gateway.



4.3.2 Análisis *de* resultados

Es esencial en la administración de redes establecer VLANs para dividir y organizar eficazmente el flujo de datos en una red. Dentro de la administración de redes, existen dos formas de configurar VLANs: la configuración puede hacerse de manera automatizada, ya sea mediante interfaces gráficas o scripts, o también a través de la configuración manual utilizando la CLI.

Automatizar la configuración de VLANs brinda una interfaz amigable y simplifica los procesos, reduciendo así el margen de error humano y agilizando la implementación del sistema. Por otro lado, la configuración mediante la interfaz de línea de comandos (CLI) permite un mayor control y flexibilidad a nivel avanzado, permitiendo a los administradores realizar ajustes precisos y análisis avanzados en las redes.

4.3.2.1 Configuración VLAN mediante CLI

Para lograr crear una configuración de una VLAN y el pool DHCP se deben tomar en cuenta los siguientes puntos:

- **Crear la VLAN**

Creación de la VLAN especificando un nombre y un ID para la VLAN.

```
#Switch: set vlans VLAN40 vlan-id 40
```

- **Configurar el VLAN Trunk**

Configuración de una interfaz como puerto troncal (trunk) para permitir múltiples VLANs:

```
#Switch: set interfaces INTERFACE_NAME unit 0 family
      ethernet-switching port-mode trunk
#Switch: set interfaces INTERFACE_NAME unit 0 family
     ethernet-switching vlan members VLAN40
```

- **Asignar Interfaces a la VLAN**

Asignación de las interfaces físicas a la VLAN. Debes especificar las interfaces que serán miembros de esta VLAN.

```
#Switch: set interfaces INTERFACE_NAME unit 0 family
     ethernet-switching vlan members VLAN40
```

- **Configurar la Interfaz IRB**

Configuración de la interfaz IRB que se asociará con la VLAN. La interfaz IRB permitirá que la VLAN se enrute y tenga una dirección IP.

```
#Switch: set interfaces irb unit 40 family inet address
                192.168.40.1/24
```

- **Asociar la Interfaz IRB con la VLAN**

  Asociar la interfaz IRB con la VLAN que has creado.

```
#Switch: set vlans VLAN40 l3-interface irb.40
```

- **Configurar el Servidor DHCP**

Configura el servidor DHCP para asignar direcciones IP desde un pool de direcciones a la VLAN.

```
#Switch: set system services dhcp-local-server group VLAN40
interface irb.40 set access address-assignment pool VLAN40
family inet network 192.168.40.0/24
#Switch: set access address-assignment pool VLAN40 family
inet range DHCP_RANGE low 192.168.40.10
#Switch: set access address-assignment pool VLAN40
family inet range DHCP_RANGE high 192.168.40.100
#Switch: set access address-assignment pool VLAN40
family inet dhcp-attributes router 192.168.40.1
```

- **Configurar DHCP Relay**

Configura el DHCP Relay para reenviar las solicitudes DHCP desde la VLAN 40 al servidor DHCP.

```
#Switch: set forwarding-options dhcp-relay group VLAN40
interface irb.40 set forwarding-options dhcp-relay group
VLAN40 server 192.168.1.1
```

4.3.2.2 Tiempo estimado con CLI

Se deben ejecutar aproximadamente 11 comandos por cada dispositivo de red, lo cual toma un tiempo estimado de 20 minutos por dispositivo. Si multiplicamos estos comandos por el número total de dispositivos de red, obtenemos un total de 66 comandos, con un tiempo estimado de 2 horas para completarlos. Estos comandos deben ser ingresados manualmente por el administrador de red, lo que aumenta significativamente la probabilidad de errores.

4.3.2.3 Tiempo estimado con Automatización

Desde la interfaz de creación para el despliegue completo de la VLAN se logra un tiempo aproximado de 1.5 minutos. Esto demuestra que la automatización reduce significativamente el tiempo necesario, de 2 horas a solo 1.5 minutos, lo que representa una reducción del 98.64%. Esta mejora no solo ahorra tiempo, sino que también minimiza los errores asociados con la entrada manual de comandos por parte del administrador de red.

Figura 28
Tiempo de ejecución creación VLAN.

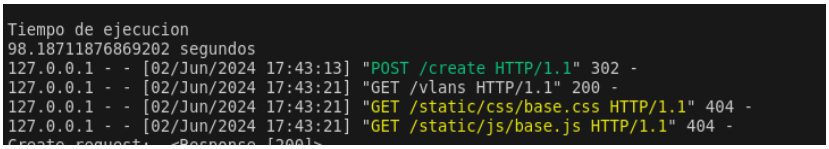


Figura 29
Tiempo de ejecución eliminación VLAN.

Tabla 1
Comparativa de Tiempos.

	CLI	**Automatización**	**Ahorro**
Crear	2 horas	98.18 segundos	98.64%
Eliminar	2,5 horas	66.7 segundos	99.26%
Leer	N/A	8 segundos	N/A

Nota. Esta tabla muestra una comparación entre el tiempo requerido para realizar el despliegue de una VLAN mediante la CLI y una interfaz de automatización.

Conclusiones Y Recomendaciones

Conclusiones

- Comparar la configuración de redes mediante la CLI tradicional con la automatización usando protocolos modernos como NETCONF y RESTCONF, junto con lenguajes de programación como Python, revela un cambio notable en la gestión de redes. Aunque la CLI ha sido el estándar durante mucho tiempo, su enfoque manual y propenso a errores puede limitar la eficiencia en redes complejas. En cambio, la automatización permite gestionar las redes de manera más eficiente y ágil reduciendo los errores.
- Automatizar con NETCONF y RESTCONF, utilizando Python, permite crear scripts que se pueden reutilizar y adaptar fácilmente, gestionando configuraciones de red de manera más rápida y precisa. Esto no solo ahorra tiempo en tareas rutinarias, sino que también mejora la consistencia y la capacidad de respuesta de la red. Además, integrar estos protocolos facilita la interoperabilidad y una gestión centralizada.
- Automatizar tareas con una interfaz puede ahorrar mucho tiempo en comparación con el uso de la CLI. Según los resultados mostrados, la automatización reduce el tiempo necesario para crear y eliminar una VLAN de varias horas a solo unos segundos. Por ejemplo, crear una VLAN ahorra un 98.64% del tiempo, y eliminarla ahorra aún más, un 99.26%. Estos ahorros de tiempo pueden hacer que las operaciones sean mucho más eficientes.

Recomendaciones

- Se sugiere utilizar GNS3 en Linux porque este sistema operativo maneja mejor los recursos del equipo. Linux destaca por su eficiencia y su capacidad para optimizar el uso de la memoria y el procesamiento, lo que proporciona un rendimiento más suave y estable al ejecutar GNS3.
- Es recomendable trabajar con el formato JSON para el procesamiento de las distintas configuraciones, ya que presenta una estructura mucho más comprensible y fácil de manejar, lo que ayuda a evitar que los scripts se vuelvan demasiado complejos.
- Es fundamental aplicar un enfoque distinto en el desarrollo de los scripts según la función de cada dispositivo, como switch core, switch de acceso y switch de distribución. Dado que desempeñan funciones diferentes, su configuración no puede ser uniforme.

Bibliografía

Chunrong, Z. (2022). *What Is NETCONF? Why Do We Need It? - Huawei*. https://info.support.huawei.com/info-finder/encyclopedia/en/NETCONF.html

Cloudflare. (n.d.). *¿Qué es una red definida por software (SDN)?* Retrieved April 13, 2024, from https://www.cloudflare.com/es-es/learning/network-layer/what-is-sdn/

Comunidad Huawei Enterprise. (2024, January 17). *¿Qué es el modelo YANG? - Conceptos Básicos | Programabilidad*. https://forum.huawei.com/enterprise/es/%C2%BFQu%C3%A9-es-el-modelo-Yang-Conceptos-B%C3%A1sicos/thread/747734100476837888-667212884033286144

Conde Gil, A. M. (2021). *Automatización de redes informáticas con Python*. http://hdl.handle.net/10609/126926

Etiris Magazine. (2023, May 20). *Network Automation with Python: A Comprehensive Guide*. https://medium.com/@etirismagazine/network-automation-with-python-a-comprehensive-guide-8ca7dc6db77d

Fortinet. (n.d.). *What is Network Automation? How Does it Work?* Retrieved April 13, 2024, from https://www.fortinet.com/resources/cyberglossary/network-automation

Hernandez, E. (2023, June 11). *01 Netmiko – La Herramienta de Automatización para Dispositivos de Red*. https://community.cisco.com/t5/blogs-general/01-netmiko-la-herramienta-de-automatizaci%C3%B3n-para-dispositivos-de/ba-p/4954431

Juniper Networks. (2023, March 21). *Junos OS Architecture Overview*. https://www.juniper.net/documentation/us/en/software/junos/junos-

overview/topics/concept/junos-software-architecture.html#junos-os-architecture-overview__id-10113981

Milios, & George. (2021). *Network Automation using Python*. https://repository.ihu.edu.gr//xmlui/handle/11544/29802

Red Hat. (2019, January 8). *¿Qué es la gestión de las redes?* https://www.redhat.com/es/topics/management/what-is-network-management

Torrejón Vázquez, D. (2024). *Desarrollo de un servicio API REST para simplificar la administración de redes multi-vendor y multi-entorno*. http://hdl.handle.net/10609/149501

Xiao, S., Jia, R., & Guo, J. (2010). Research and implement on compatibility and scalability of agent of platform for network management based on NETCONF protocol. *2010 International Conference on Measuring Technology and Mechatronics Automation, ICMTMA 2010*, *2*, 1035–1038. https://doi.org/10.1109/ICMTMA.2010.705

ANEXOS

Anexo 1: Repositorio de proyecto

https://github.com/jeremymore/APPredes