DESARROLLO DE REDES PROGRAMABLES CON PYTHON PARA LA CONFIGURACIÓN DEL ENRUTAMIENTO OSPFv2

DEVELOPMENT OF PROGRAMMABLE NETWORKS WITH PYTHON FOR OSPFv2 ROUTING CONFIGURATION

Sornoza Púa Gregory Iván [1*]

[1] Estudiante de la carrera Ingeniería en Tecnologías de la Información. Facultad de Ingeniería Civil, Ambiental y de Tecnologías de la Información de la Universidad Técnica de Machala, Ecuador. ORCID: https://orcid.org/0009-0004-0640-1566. Correo: gsornoza1@utmachala.edu.ec

Ronald Richard Morocho Morán [2]

[2] Estudiante de la carrera Ingeniería en Tecnologías de la información. Facultad de Ingeniería Civil, Ambiental y de Tecnologías de la Información de la Universidad Técnica de Machala, Ecuador ORCID: https://orcid.org/0009-0007-8528-4629. Correo: rmorocho6@utmachala.edu.ec

Mariuxi Paola Zea Ordoñez [3]

[3] Magister en Sistemas de Información Gerencial. Docente en la Carrera Tecnologías de la Información de la Facultad de Ingeniería Civil, Ambiental y Tecnologías de la Información de la Universidad Técnica de Machala, Ecuador. ORCID: https://orcid.org/0000-0001-8860-6282. Correo: mzea@utmachala.edu.ec

Oscar Efrén Cárdenas Villavicencio [4]

[4] Magister en Telecomunicaciones. Docente en la Carrera Tecnologías de la Información de la Facultad de Ingeniería Civil, Ambiental y Tecnologías de la Información de la Universidad Técnica de Machala, Ecuador. ORCID: https://orcid.org/0000-0001-6570-8040. Correo: oecardenas@utmachala.edu.ec

*** Autor para correspondencia: gsornoza1@utmachala.edu.ec**

Resumen

El aprovisionamiento manual de protocolos de enrutamiento motivó esta investigación debido al tiempo requerido y a las incidencias de configuración. El objetivo fue desarrollar y evaluar un controlador basado en Python para automatizar el aprovisionamiento del protocolo Open Shortest Path First versión 2 en una topología Spine-Leaf. El desarrollo integró Scrum y principios de NetDevOps, mientras que la evaluación siguió un diseño experimental comparativo con enfoque cuantitativo. El controlador incorporó

Esta obra está bajo una licencia *Creative Commons* de tipo (CC-BY-NC-SA).

Grupo Editorial "ALEMA-Pentaciencias" E-mail: alema.pentaciencias@gmail.com

descubrimiento de dispositivos, inventario dinámico, identificación de enlaces, despliegue de configuraciones y validación automática. En un entorno emulad con cinco routers Cisco se realizaron veinte ejecuciones manuales y veinte automatizadas, restableciendo las mismas condiciones iniciales antes de cada prueba. El tiempo medio de despliegue disminuyó de 486,00 a 8,17 segundos (98,32 %), mientras que el tiempo total de aprovisionamiento pasó de 486,00 a 89,15 segundos (81,66 %). Las pruebas manuales registraron 53 incidencias, frente a ninguna en las automatizadas. Ambos métodos finalizaron con el enrutamiento operativo. Estos resultados indican que la automatización redujo los tiempos y las incidencias del aprovisionamiento en las condiciones evaluadas.

Palabras clave: automatización de redes; OSPFv2; Python; Spine-Leaf; aprovisionamiento de redes

Abstract

The time and configuration incidents associated with manual routing provisioning motivated this study. The objective was to develop and evaluate a Python-based controller to automate Open Shortest Path First version 2 provisioning in a Spine-Leaf topology. Development combined Scrum and NetDevOps principles, while evaluation followed a quantitative comparative experimental design. The controller integrated device discovery, a dynamic inventory, link identification, configuration deployment, and automated validation. In an emulated environment with five Cisco Routers, twenty manual and twenty automated runs were performed, restoring the same initial conditions before each trial. Mean deployment time decreased from 486.00 to 8.17 seconds (98.32%), whereas total provisioning time decreased from 486.00 to 89.15 seconds (81.66%). Manual runs recorded 53 incidents, compared with none in automated runs. Both methods ultimately achieved operational routing. These results indicate that automation reduced provisioning times and configuration incidents under the evaluated conditions.

Keywords: *network automation; OSPFv2; Python; Spine-Leaf; network provisioning*

Fecha de recibido: 23/07/2026
Fecha de aceptado: 26/10/2026
Fecha de publicado: 18/11/2026

Introducción

La administración de infraestructuras de red requiere procedimientos que permitan mantener configuraciones consistentes y reducir los tiempos asociados con las tareas operativas. Sin embargo, la gestión tradicional mediante interfaces de línea de comandos (CLI) depende de la intervención directa del administrador, especialmente cuando es necesario aplicar configuraciones similares en múltiples dispositivos. Esta práctica incremente el tiempo requerido para el aprovisionamiento y puede ocasionar errores durante la ejecución de instrucciones (Adewa et al., 2025; Yulán et al., 2024). En infraestructuras de mayor complejidad, dichas dificultades adquieren especial relevancia, lo que ha motivado el desarrollo de técnicas de automatización

Esta obra está bajo una licencia *Creative Commons* de tipo (CC-BY-NC-SA).
Grupo Editorial "ALEMA-Pentaciencias" E-mail: alema.pentaciencias@gmail.com

orientadas a simplificar la administración y reducir la dependencia de procedimientos manuales (Abdellatif et al., 2026).

Entre las tareas de administración se encuentre la configuración de protocolos de enrutamiento dinámico, como Open Shortest Path First versión2 (OSPFv2), utilizado para determinar rutas de comunicación mediante información del estado de los enlaces. Su provisionamiento requiere habilitar el protocolo en los dispositivos correspondientes, establecer los parámetros de enrutamiento y verificar la formación de adyacencias. Estas operaciones adquieren particular importancia en arquitectura Spine-Leaf, donde la interconexión entre los niveles de la topología proporciona múltiples caminos de comunicación. Aunque el protocolo permite calcular rutas y responder a cambios en la red, su configuración inicial continúa representando una actividad administrativa que puede beneficiarse de mecanismos programáticos.

El desarrollo de redes programables ha impulsado alternativas para automatizar estas operaciones. Las redes definidas por software (Software-Defined Networking, SDN) constituyen uno de los enfoques que han contribuido a la evolución de la administración de infraestructuras mediante la separación lógica entre los planos de control y datos (Anerousis et al., 2021). De forma complementaria, la automatización mediante Python permite gestionar dispositivos de red a través de conexiones remotas y ejecutar configuraciones sin introducir individualmente los comandos en cada equipo. Mazin et al. (2021) identificaron mejoras en los tiempos de configuración y una reducción de errores asociados con tarea administrativas repetitivas mediante este lenguaje. Estas capacidades pueden integrarse en herramientas de orquestación que utiliza protocolos de administración como Secure Shell (SHH) y bibliotecas especializadas, entre ellas Netmiko.

Las investigaciones recientes han examinado la automatización desde diferentes perspectivas. Osei-Wusu et al. (2025) reportaron reducciones superiores al 99 % en el tiempo de ejecución de respaldos automatizados sobre dispositivos Cisco mediante Python y Netmiko. En el ámbito del enrutamiento, Tao et al. (2021) estudiaron la integración de SDN con OSPF mutiárea, mientras que Viuche et al. (2025) abordaron el aprovisionamiento dinámico de arquitecturas Leaf-Spine mediante mecanismos Zero-Touch. Aunque estos trabajos aportan soluciones para la administración y operación de redes, sus enfoques difieren del aprovisionamiento automatizado de OSPFv2 mediante descubrimiento topológico y configuración programática de dispositivos Cisco.

La incorporación de prácticas NetDevOps también ha contribuido el desarrollo de procedimientos de automatización reproducibles. Dsilva et al. (2025) analizaron su aplicación en redes ópticas, destacando la utilidad de integrar la automatización y la validación de configuraciones en las operaciones de red. No obstante, entre los antecedentes examinados no se identifica una evaluación directamente equivalente que integre el descubrimiento de dispositivos, la identificación de enlaces mediante CDP, el recorrido topológico mediante BFS y el despliegue automatizado de OSPFv2 en una arquitectura Spine-Leaf, comparando sus resultados con la configuración manual bajo condiciones controladas. Esta diferencia entre los enfoques existentes y el procedimiento propuesto delimita el vacío de conocimiento del presente estudio, centro en evaluar el efecto de dicha integración sobre los tiempos de aprovisionamiento y las incidencias de configuración.

En este contexto, se formuló la siguiente pregunta de investigación: ¿La automatización del aprovisionamiento de OSPFv2 en una topología Spine-Leaf mediante un controlador basado en Python reduce

Esta obra está bajo una licencia *Creative Commons* de tipo (CC-BY-NC-SA).

Grupo Editorial "ALEMA-Pentaciencias" E-mail: alema.pentaciencias@gmail.com

los tiempos operativos y las incidencias de configuración en comparación con el procedimiento manual? Se planteó como hipótesis que el uso del controlador permitiría disminuir el tiempo de despliegue, el tiempo total de aprovisionamiento y las incidencias de configuración respecto al método tradicional basado en CLI.

El objetivo de la investigación fue desarrollar y evaluar un controlador basado en Python para automatizar el aprovisionamiento de OSPFv2 en una topología Spine-Leaf, integrando mecanismos de descubrimiento topológico, generación de configuraciones y validación funcional. Para comprobar su desempeño, se adoptó un enfoque cuantitativo con un diseño experimental comparativo, desarrollado en GNS3 mediante veinte ejecuciones manuales y veinte automatizadas bajo condiciones iniciales equivalentes. La evolución considero los tiempos de despliegue, convergencia posterior y aprovisionamiento total, además de las incidencias registradas y la proporción de ejecuciones completadas sin incidencias.

Materiales y métodos

Diseño metodológico de la investigación

La investigación se desarrolló mediante un enfoque cuantitativo, con un diseño experimental comparativo orientado a evaluar el desempeño del aprovisionamiento automatizado del protocolo Open Shortest Path Fist versión 2 (OSPFv2) frente al procedimiento tradicional de configuración manual. La evaluación se sustentó en la medición de los tiempos de despliegue, convergencia y aprovisionamiento total, así como en el registro de incidencias de configuración y proporción de ejecuciones completadas sin incidencias.

Se establecieron dos escenarios experimentales sobre una misma topología Spine-Leaf. El primero consistió en la configuración manual del protocolo mediante la interfaz de línea de comandos de los dispositivos Cisco, mientras que el segundo utilizó un controlador desarrollado en Python para ejecutar automáticamente el descubrimiento de la infraestructura, la generación de configuraciones y el despliegue de OSPFv2. Ambos escenarios fueron sometidos a veinte ejecuciones independientes, obteniéndose un total de cuarenta pruebas bajo condiciones controladas.

El diseño permitió examinar las diferencias entre los dos métodos de aprovisionamiento, manteniendo constantes las características de la infraestructura y utilizando indicadores comunes para determinar la eficiencia y confiabilidad del proceso.

Entorno experimental y topología de red

La implementación y evaluación del controlador se realizaron en una máquina virtual ejecutada mediante VMware Workstation 17 Pro, con Ubuntu Server 20.04.6 LTS como sistema operativo. El entorno contó con dos procesadores virtuales, 4 GB de memoria RAM y 80 GB de almacenamiento. Para la emulación de la infraestructura se utilizó GNS3 versión 2.2.54 con imágenes Cisco IOS c7200, mientras que el controlador fue desarrollado en Python 3.8.10 y empleó Netmiko 4.4.0 para establecer conexiones SSH y ejecutar configuraciones remotas.

La infraestructura experimental estuvo conformada por cinco routers Cisco distribuidos en una arquitectura Spine-Leaf. La capa Spine estuvo integrada por los dispositivos R1 y R2, mientras que R3, R4 y R5 conformaron la capa Leaf. Cada router Leaf mantuvo enlaces hacia ambos dispositivos Spine, proporcionando

Esta obra está bajo una licencia *Creative Commons* de tipo (CC-BY-NC-SA).

Grupo Editorial "ALEMA-Pentaciencias" E-mail: alema.pentaciencias@gmail.com

conectividad redundante entre los niveles de la topología. Esta disposición permitió implementar OSPFv2 sobre múltiples enlaces y verificar el establecimiento de las relaciones de vecindad correspondientes.

La infraestructura se organizó mediante una separación entre el plano de administración y el plano de datos. El primero proporcionó conectividad de gestión independiente para el acceso remoto a los dispositivos mediante SSH, mientras que el segundo comprendió los enlaces utilizados para el intercambio de información de enrutamiento y el tráfico de producción. La configuración de OSPFv2 se limitó a una única área, identificada como Área 0.

La distribución de los dispositivos, sus interconexiones y la separación de los segmentos de red se presentan en la Figura 1.

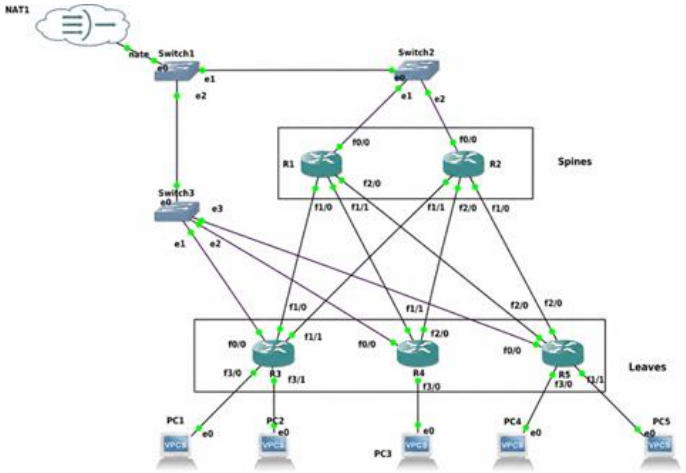


Figura 1. Topología Spine-Leaf utilizada durante la experimentación

Desarrollo del controlador mediante Scrum y NetDevOps

El desarrollo del controlador se organizó mediante Scrum, utilizando ciclos incrementales para planificar e implementar sus funcionalidades (Schwaber & Sutherland, 2020; Sassa et al., 2023). De manera complementaria, se incorporaron principios NetDevOps relacionados con la automatización programática, la reproducibilidad de configuraciones y la validación de la infraestructura (Shah & Dubaria, 2019; Dsilva et al., 2025). Scrum proporcionó la estructura para gestionar el desarrollo del software, mientras que NetDevOps orientó las decisiones técnicas necesarias para automatizar las operaciones de red. La relación entre ambos enfoques y su contribución al controlador se presenta en la Figura 2.

Esta obra está bajo una licencia *Creative Commons* de tipo (CC-BY-NC-SA).

Grupo Editorial "ALEMA-Pentaciencias" E-mail: alema.pentaciencias@gmail.com

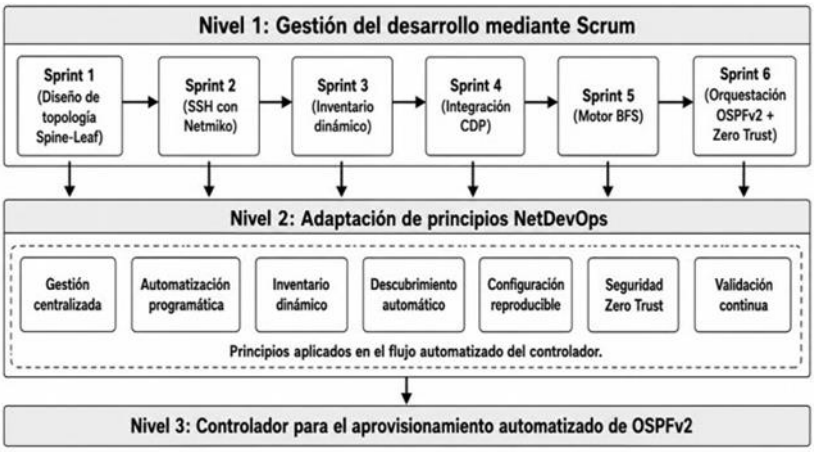


Figura 2. Integración entre Scrum y NetDevOps durante el desarrollo del controlador.

El proceso se distribuyó en seis sprints técnicos, definidos a partid de las funcionalidades necesarias para construir progresivamente la solución. Cada sprint incorporó un componente que posteriormente fue integrado al flujo completo de aprovisionamiento. La distribución de estas actividades se muestra en la Figura 3.

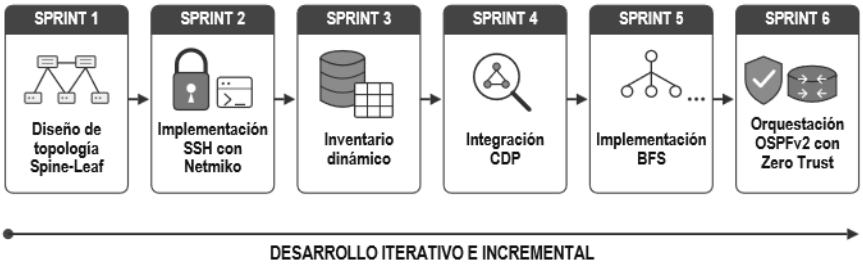


Figura 3. Adaptación de Scrum al desarrollo del controlador propuesto.

Sprint 1. Diseño de la topología Spine-Leaf. Se diseñó y construyó la infraestructura experimental mediante la distribución de los cinco routers en los niveles Spine y Leaf. Se definieron las conexiones entre los dispositivos, el direccionamiento IP y la separación del plano de administración respecto al plano de datos. Además, se establecieron las condiciones iniciales necesarias para la implementación posterior de OSPFv2. Esta etapa proporcionó la infraestructura sobre la cual se desarrollaron y comprobaron las funcionalidades del controlador.

Sprint 2. Implementación de la conectividad SSH. Se desarrolló el mecanismo de comunicación entre el controlador y los dispositivos Cisco mediante el protocolo Secure Shell (SSH), utilizando la biblioteca Netmiko. Esta implementación permitió establecer sesiones remotas, autenticar las conexiones, ejecutar comandos de administración y recuperar las respuestas generadas por los routers, También se incorporó el manejo de excepciones para identificar posibles errores de autenticación, problemas de conectividad o indisponibilidad de dispositivos, proporcionando una base de comunicación para las siguientes funciones de automatización.

Esta obra está bajo una licencia *Creative Commons* de tipo (CC-BY-NC-SA).

Grupo Editorial "ALEMA-Pentaciencias" E-mail: alema.pentaciencias@gmail.com

Sprint 3. Desarrollo del inventario dinámico. Se implementó un módulo encargado de registrar automáticamente los dispositivos identificados durante el descubrimiento de la infraestructura. La información se organizó en un inventario temporal almacenado en memoria RAM, que contenía nombres de host, direcciones IP de administración, interfaces disponibles y otros parámetros necesarios para la ejecución de configuraciones. Este procedimiento permitió utilizar información obtenida durante la ejecución del controlador, evitando depender de un archivo de inventario estático que requiera modificaciones manuales antes de cada despliegue.

Sprint 4. Integración del protocolo CDP. Se incorporó Cisco Discovery Protocol (CDP) para identificar automáticamente las relaciones de vecindad existentes entre los dispositivos. Mediante consultas remotas ejecutadas por SSH, el controlador recuperó la información correspondiente a los routers vecinos y las interfaces locales y remotas utilizadas en sus interconexiones. Los datos obtenidos permitieron construir una representación lógica de la infraestructura, necesaria para determinar los dispositivos y enlaces que participarían posteriormente en el aprovisionamiento de OSPFv2.

Sprint 5. Implementación del algoritmo BFS. A partir de la información topológica recopilada mediante CDP, se desarrolló un mecanismo de exploración basado en Breadth-First Search (BFS). El algoritmo recorrió la infraestructura por niveles y utilizó un registro de los dispositivos previamente procesados para evitar visitas redundantes y configuraciones duplicadas. Como resultado, el controlador obtuvo una secuencia organizada de los routers identificados, que sirvió como referencia para ejecutar las etapas posteriores del aprovisionamiento automatizado

Sprint 6. Orquestación de OSPFv2 y aislamiento del plano de administración. Se integraron los módulos desarrollados para construir el orquestador encargado de generar y aplicar las configuraciones del protocolo OSPFv2. El sistema identificó las interfaces correspondientes al plano de datos y preparó las instrucciones necesarias para habilitar el protocolo en el Área 0. Adicionalmente, se incorporó un criterio de aislamiento inspirado en Zero Trust, mediante el cual se excluyó automáticamente la interfaz de administración del dominio de enrutamiento (Ren et al., 2025). Finalmente, se integraron procedimientos de validación para comprobar el establecimiento de las adyacencias, la disponibilidad de rutas y la conectividad entre los dispositivos.

La finalización de los seis sprints permitió disponer de un controlador que integró las funcionalidades de descubrimiento, identificación topológica, generación de configuraciones, despliegue y validación. Su implementación se sustentó en los procedimientos desarrollados durante cada etapa, sin requerir la definición manual de un inventario completo de dispositivos antes de ejecutar el aprovisionamiento.

Arquitectura lógica y funcionamiento del controlador

El controlador se organizó mediante un conjunto de componentes encargados de ejecutar secuencialmente las operaciones necesarias para aprovisionar OSPFv2. Su funcionamiento comenzó con el descubrimiento de los dispositivos accesibles a través de la red de administración y finalizó con la comprobación de las configuraciones aplicadas. El flujo operativo adaptado a los principios NetDevOps se presenta en la Figura 4.

Esta obra está bajo una licencia *Creative Commons* de tipo (CC-BY-NC-SA).

Grupo Editorial "ALEMA-Pentaciencias" E-mail: alema.pentaciencias@gmail.com

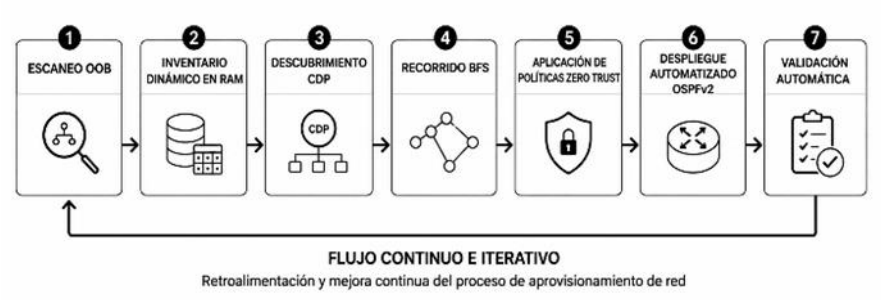


Figura 4. Flujo NetDevOps adaptado al aprovisionamiento automatizado de OSPFv2.

El procedimiento inició con un escaneo Out-of-Band (OOB) sobre una red de gestión para localizar los dispositivos disponibles. A partir de la información obtenida, el controlador construyó un inventario dinámico en memoria RAM que reunió los parámetros requeridos para establecer conexiones y ejecutar instrucciones sobre los routers identificados.

Posteriormente, mediante consultas CDP, se recuperaron las relaciones de vecindad y las interfaces que conectaban los dispositivos. Esta información fue procesada para construir una representación lógica de la topología, sobre la cual se aplicó BFS con el propósito de determinar una secuencia de recorrido que evitara el procesamiento repetido de un mismo equipo.

Antes de cada despliegue, el controlador identificó las interfaces destinadas a tareas administrativas y las excluyó de la configuración del protocolo de enrutamiento. Esta operación mantuvo la red de gestión fuera del dominio OPSFv2 y evitó que las interfaces administrativas participaran en el intercambio de información de enrutamiento. A continuación, el orquestador generó las instrucciones necesarias para habilitar OSPFv2 en el Área 0 y las aplicó a los dispositivos mediante sesiones SSH establecidas con Netmiko.

La última etapa consistió en verificar el funcionamiento de la infraestructura configurada. Para ello, el controlador comprobó el establecimiento de las adyacencias OSPS, la disponibilidad de rutas aprendidas y la conectividad extremo a extremo. Asimismo, se contempló el registro de posibles errores de autenticación, fallos de configuración e inconsistencias detectadas durante la ejecución.

La interacción de estos componentes con la infraestructura Spine-Leaf, así como la separación entre la lógica de orquestación y los dispositivos administrados, se representa en la Figura 5.

Esta obra está bajo una licencia *Creative Commons* de tipo (CC-BY-NC-SA).

Grupo Editorial "ALEMA-Pentaciencias" E-mail: alema.pentaciencias@gmail.com

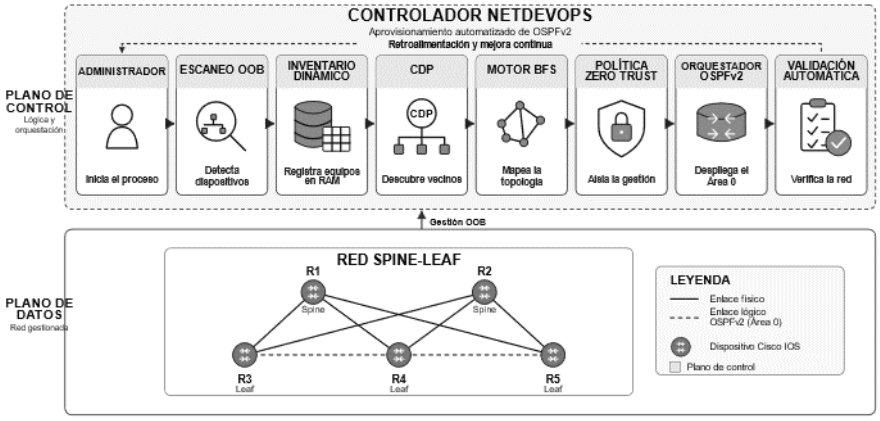


Figura 5. Arquitectura lógica del controlador propuesto.

Proceso experimental

La evaluación se realizó comparando dos procedimientos de aprovisionamiento ejecutados sobre la misma infraestructura. En el escenario manual, un profesional especializado en administración de redes configuró OSPFv2 mediante comandos introducidos directamente en las interfaces CLI de los routers. En el escenario automatizado, el controlador desarrollado en Python ejecutó las operaciones de descubrimiento, generación y aplicación de configuraciones sin requerir la introducción individual de los comandos en cada dispositivo.

Antes de cada ejecución, se eliminó la configuración de OSPFv2 de los routers para restablecer las condiciones iniciales del experimento. Se mantuvieron constantes la topología el direccionamiento IP, los dispositivos utilizados y los recursos asignados al entorno virtual. Este procedimiento se aplicó en ambos escenarios para asegurar que las pruebas partieran de una infraestructura equivalente y que la diferencia principal correspondiera al método de aprovisionamiento empleado.

Se realizaron veinte ejecuciones independientes por cada procedimiento, con un total de cuarenta pruebas. Durante cada repetición se registraron los tiempos asociados con la configuración del protocolo y el establecimiento de las adyacencias, así como las incidencias que requirieron intervenciones adicionales. Una vez concluido el aprovisionamiento, se comprobó el funcionamiento del protocolo mediante la revisión de las relaciones de vecindad, las rutas aprendidas y la conectividad entre los dispositivos.

Variables e indicadores de evaluación

La variable independiente del estudio fue el método de aprovisionamiento de OSPFv2, establecido en dos modalidades: manual y automatizado. Las variables dependientes comprendieron los tiempos de despliegue, convergencia posterior y aprovisionamiento total, junto con incidencias de configuración y la tasa de ejecuciones sin incidencias. Los indicadores y sus respectivas unidades se presentan en la Tabla 1.

Tabla 1. Variables e indicadores de evaluación

Variable	Tipo	Indicador	Unidad
Método de aprovisionamiento	Independiente	Manual o automatizado	Modalidad
Tiempo de despliegue	Dependiente	Duración de la configuración	Segundos

Esta obra está bajo una licencia *Creative Commons* de tipo (CC-BY-NC-SA).

Grupo Editorial "ALEMA-Pentaciencias" E-mail: alema.pentaciencias@gmail.com

Tiempo de convergencia	Dependiente	Establecimiento completo de adyacencias OSPFv2	Segundos
Tiempo total de aprovisionamiento	Dependiente	Duración desde el inicio hasta el estado FULL	Segundos
Incidencias de configuración	Dependiente	Número de eventos que requirieron corrección	Cantidad
Tasa de ejecuciones sin incidencias	Dependiente	Proporción de pruebas sin incidencias	Porcentaje

Medición y análisis de los datos

Los tiempos experimentales se registraron en formato minutos, centésimas y posteriormente se convirtieron a segundo para facilitar su procesamiento. Se establecieron criterios comunes de medición para ambos métodos, considerando las diferencias operativas entre la configuración secuencial realizada por el administrador y la ejecución programática del controlador.

El tiempo de despliegue correspondió al intervalo transcurrido desde el inicio del proceso de configuración hasta la finalización de la aplicación de OSPFv2 en todos los dispositivos. En el escenario manual, el registro comenzó con la introducción del primer comando de configuración en el primer router y terminó después de ejecutar el último comando en el dispositivo final. En el escenario automatizado, se consideró el intervalo comprendido entre la ejecución del controlador y la finalización del despliegue automático de las configuraciones sobre los equipos descubiertos.

El tiempo de convergencia posterior al despliegue se definió como el intervalo transcurrido entre la finalización de la configuración y el establecimiento completo de las adyacencias OSPFv2 previstas en la topología. Su verificación se realizó mediante el comando show ip neighbor, considerando que la infraestructura había completado esta fase cuando todas las relaciones de vecindad esperadas presentaban el estado FULL.

Debido a la configuración secuencial de los routers en el escenario manual, algunas adyacencias podían establecerse mientras el administrador continuaba trabajando sobre los dispositivos restantes. Por esta razón, el tiempo de convergencia posterior podía ser igual a cero si todas las relaciones ya habían alcanzado el estado FULL al terminal la configuración. Esta condición no representaba una convergencia instantánea, sino la superposición entre el despliegue manual y el establecimiento de las adyacencias.

Para comprar la duración completa de ambos procedimientos se calculó el tiempo total de aprovisionamiento, definido como el intervalo desde el inicio de la configuración hasta que todas las adyacencias OSPFv2 alcanzaron el estado FULL. Esta métrica se determinó mediante la Ecuación (1):

$$TTA = TD + TCPD \qquad (1)$$

Donde TTA representa el tiempo total de aprovisionamiento, TD corresponder al tiempo de despliegue y $TCPD$ al tiempo de convergencia posterior al despliegue. Esta medición permitió comparar integralmente la duración necesaria para dejar operativa la infraestructura de enrutamiento.

Las incidencias de configuración comprendieron los errores de sintaxis en los comandos, los fallos de conectividad y las adyacencias OSPFv2 incompletas que requirieron una acción correctiva. Cada incidencia

Esta obra está bajo una licencia *Creative Commons* de tipo (CC-BY-NC-SA).

Grupo Editorial "ALEMA-Pentaciencias" E-mail: alema.pentaciencias@gmail.com

se contabilizó incluso cuando pudo resolverse durante la misma ejecución, debido a que representaba una intervención adicional dentro del proceso de aprovisionamiento.

La tasa de ejecuciones sin incidencias se definió como el porcentaje de pruebas que completaron el aprovisionamiento sin presentar ninguna de las incidencias establecidas. Una ejecución se consideró libre de incidencias cuando no requirió correcciones de comandos, intervenciones adicionales ni ajustes para completar las adyacencias OSPFv2. Esta métrica se calculó mediante la Ecuación (2):

$$TSE = \frac{ESE}{ET} \times 100 \qquad (2)$$

Donde TSE representa la tasa de ejecuciones sin incidencias, ESE corresponde al número de ejecuciones que no registraron incidencias y ET al número total de ejecuciones realizadas en cada escenario.

De manera complementaria, se verificó el existo funcional de cada prueba mediante la comprobación de las adyacencias en estado FULL, la disponibilidad de rutas OSPFv2 y la conectividad extremo a extremo. Esta comprobación permitió determinar si ambos procedimientos lograron dejar correctamente operativa la infraestructura, independientemente de las incidencias presentadas durante su configuración.

Los datos recopilados se analizaron mediante estadística descriptiva, utilizando medias, desviaciones estándar y rangos de los tiempos registrados, así como el número de incidencias y los porcentajes de ejecuciones sin incidencias. Para cuantificar las diferencias de duración entre los escenarios, se calcularon las reducciones porcentuales tomando como referencia los promedios obtenidos mediante el procedimiento manual. Este análisis permitió contrastar el desempeño temporal y la confiabilidad de ambos métodos en las condiciones experimentales establecidas.

Resultados y discusión

Desempeño temporal del aprovisionamiento

La evaluación comprendió cuarenta ejecuciones experimentales, distribuidas equitativamente entre los procedimientos manual y automatizado. Se compararon los tiempos de despliegue de OPSFv2, convergencia posterior y aprovisionamiento total. La Tabla 2 presenta las medidas y desviación estándar (DE) obtenidas.

Tabla 2. Comparación de los tiempos registrados en los escenarios evaluados.

Métrica	Manual (media ± DE)	Automatizado (media ± DE)	Reducción
Tiempo de despliegue	486,00 ± 118,69 s	8,17 ± 1,17 s	98,32 %
Convergencia posterior al despliegue	0,00 ± 0,00 s	80,98 ± 6,01 s	No comparable aisladamente
Tiempo total de aprovisionamiento	486,00 ± 118,69 s	89,15 ± 6,94 s	81,66 %

El despliegue manual requirió un promedio de 486,00 segundos, con valores entre 306,68 y 660,67 segundos, mientras que el automatizado necesitó 8,17 segundos, con registros entre 5,68 y 10,83 segundos. Esta diferencia representó una reducción del 98,32 %, equivalente a un procedimiento manual aproximadamente 59 veces más lento. La menor desviación estándar del escenario automatizado también evidenció una ejecución más uniforme.

Esta obra está bajo una licencia *Creative Commons* de tipo (CC-BY-NC-SA).

Grupo Editorial "ALEMA-Pentaciencias" E-mail: alema.pentaciencias@gmail.com

La convergencia posterior presento un comportamiento diferente. En el procedimiento manual se registraron 0,00 segundos porque las adyacencias OSPFv2 se establecieron mientras el operador continuaba configurando los dispositivos restantes. En el automatizado, el promedio fue de 80,98 segundos, con valores entre 68,67 y 95,73 segundos. Esta diferencia respondió a la secuencia operativa de ambos métodos y no significo que OSPFv2 convergiera instantáneamente durante la configuración manual.

Al considerar el proceso completo hasta alcanzar todas las adyacencias en estado FULL, el tiempo total de aprovisionamiento disminuyo de 486,00 a 89,15 segundos, equivalente a una reducción del 81,66 %. Incluso al comparar la ejecución manual más rápida (306,68 segundos) con la automatizada más lenta (106,56 segundos), se mantuvo una reducción del 65,25 %, confirmando la ventaja temporal del controlador dentro de las condiciones evaluadas.

Incidencias de configuración y ejecuciones sin incidencias

Además de los tiempos, se registraron las incidencias que requirieron intervenciones correctivas y se verificó el funcionamiento de OSPFv2. Los resultados se presentan en la Tabla 3.

Tabla 3. Comparación de las incidencias registradas en los escenarios evaluados.

Indicador	Manual	Automatizado
Total de incidencias registradas	53	0
Promedio de incidencias por ejecución	2,65 ± 1,66	0,00
Intervalo de incidencias por ejecución	0-5	0
Ejecuciones sin incidencias	2 de 20	20 de 20
Tasa de ejecuciones sin incidencias	10 %	100 %
Ejecuciones funcionalmente exitosas	20 de 20	20 de 20

Durante las veinte ejecuciones manuales se contabilizaron 53 incidencias, con un promedio de 2,65 por ejecución. Únicamente dos pruebas finalizaron sin correcciones; las dieciocho restantes requirieron alguna intervención durante la introducción de comandos o la configuración de los dispositivos. Por lo contrario, las veinte ejecuciones automatizadas concluyeron sin incidencias.

Ambos procedimientos alcanzaron el mismo resultado funcional: las cuarenta pruebas finalizaron con OSPFv2 operativo, adyacencias en estado FULL, rutas aprendidas disponibles y conectividad extremo a extremo. Por tanto, la ventaja del controlador consistió en completar el aprovisionamiento con menor tiempo, variabilidad e intervención correctiva, no en conseguir un funcionamiento que el método manual no pudiera alcanzar.

Discusión de los resultados

La reducción del tiempo de despliegue concuerda con los hallazgos de Osei-Wusu et al. (2025), quienes reportaron disminuciones superiores al 99 % al automatizar respaldos de dispositivos Cisco mediante Python y Netmiko. Asimismo, Abdellatif et al. (2026) identificaron ventajas temporales de las técnicas de automatización en redes IP de alta densidad. Aunque los resultados siguen una tendencia similar, los porcentajes no son directamente equivalentes debido a las diferencias entre tareas, infraestructuras y condiciones experimentales.

Esta obra está bajo una licencia *Creative Commons* de tipo (CC-BY-NC-SA).

Grupo Editorial "ALEMA-Pentaciencias" E-mail: alema.pentaciencias@gmail.com

Respecto a las incidencias, Mazin et al. (2021) señalaron que la automatización mediante Python puede reducir errores asociados con procedimientos administrativos repetitivos. La ausencia de incidencias en el escenario automatizado también se relaciona con las prácticas NetDevOps estudiadas por Dsilva et al. (2025), especialmente la reproducibilidad y validación de configuraciones. En esta investigación, dichas características se reflejaron en la menor dispersión de los tiempos y la ejecución uniforme del procedimiento.

Desde la perspectiva del enrutamiento, Viuche et al. (2025) compararon el desempeño de redes definidas por software y redes IP convencionales mediante protocolos de enrutamiento interior. A diferencia de este estudio, la presente investigación evaluó el aprovisionamiento de OPSFv2 sobre dispositivos Cisco, sin sustituir el funcionamiento distribuido del protocolo. Por su parte, Miswar et al. (2023) analizaron mediante GNS3 el rendimiento de OSPF y OSPF-MPLS en la transmisión de tráfico TCP y UDP. En contraste, la evaluación desarrollada se concentró en las adyacencias, rutas aprendidas y conectividad, sin medir latencia, *jitter* ni pérdida de paquetes. Por ello, sus resultados no permiten establecer mejores en la calidad de servicio del tráfico.

La menor variabilidad del escenario automatizado también representa una ventaja operativa. Mientras que la configuración manual dependió de la introducción de comandos y de las correcciones realizadas, el controlador ejecutó un procedimiento programático consistente. Durante las pruebas manuales se observó una disminución progresiva de los tiempos, lo que sugiere un posible efecto de aprendizaje del operador. Sin embargo, la ejecución manual más rápida continuó siendo más lenta que la automatizada más prolongada.

La diferencia en los tiempos de convergencia posterior tampoco implica un desempeño superior del protocolo durante la configuración manual. En este escenario, las adyacencias se establecieron durante el despliegue secuencial; por ello, el tiempo total de aprovisionamiento constituyó el indicador más adecuado para comparar ambos procedimientos. Su reducción del 81,66 % confirmó la ventaja temporal del controlador al considerar el proceso completo.

Los resultados permiten responder afirmativamente a la pregunta de investigación: la automatización mediante Python redujo los tiempos operativos y las incidencias de configuración de OSPFv2 en la topología *Spine-Leaf* evaluada. Asimismo, la hipótesis quedó respaldada de manera descriptiva, puesto que disminuyeron el tiempo de despliegue, el tiempo total de aprovisionamiento y las incidencias. Esta valoración se fundamente en los indicadores de las cuarenta ejecuciones y no constituye una demostración de significancia estadística mediante pruebas inferenciales.

E aporte principal radica en integrar el descubrimiento topológico mediante CDP, el recorrido BFS y la configuración programática de OSPFv2 en un mismo procedimiento. Esta integración mejoró la eficiencia y consistencia del aprovisionamiento frente al método manual, dentro de las condiciones experimentales establecidas.

Conclusiones

La investigación permitió desarrollar y evaluar un controlador basado en Python para automatizar el aprovisionamiento de OSPFv2 en una topología *Spine-Leaf*. La integración del descubrimiento topológico mediante CDP, el recorrido BSF y la configuración programática de los dispositivos permitió centralizar las tareas de aprovisionamiento y validación del protocolo. Asimismo, la exclusión de las interfaces

Esta obra está bajo una licencia *Creative Commons* de tipo (CC-BY-NC-SA).

Grupo Editorial "ALEMA-Pentaciencias" E-mail: alema.pentaciencias@gmail.com

administrativas del dominio de enrutamiento mantuvo la separación entre la red de gestión y el plano de datos, cumpliendo con las funcionalidades previstas para el controlador.

La comparación experimental de veinte ejecuciones manuales y veinte automatizadas evidenció una reducción del 98,32 % en el tiempo de despliegue y del 81,66 % en el tiempo total de aprovisionamiento. Mientras que el procedimiento manual registró 53 incidencias, el automatizado completó todas las pruebas sin requerir correcciones y presentó una menor variabilidad temporal. Ambos métodos consiguieron establecer correctamente OSPFv2, pero la automatización permitió alcanzar este resultado con mayor eficiencia y consistencia. Estos hallazgos responder afirmativamente a la pregunta de investigación y respaldan descriptivamente la hipótesis planteada, dentro de las condiciones experimentales evaluadas.

El estudio estuvo limitado a una infraestructura emulada mediante GNS3, conformada por cinco routers Cisco u una única área OSPFv2, sin contemplar cambios topológicos ni escenarios de recuperación ante fallos de enlaces. Por tanto, los resultados no pueden generalizarse directamente a redes físicas o de mayor escala. Como trabajo futuro, se propone evaluar el controlador sobre dispositivos reales, ampliar las pruebas a topologías con más nodos y múltiples áreas OSPFv2, y analizar su comportamiento ante fallos y modificaciones de la infraestructura. Estas evaluaciones permitirán determinar su capacidad de adaptación y escalabilidad en condiciones operativas más diversas.

Agradecimientos

Los autores agradecen a la Universidad Técnica de Machala por el respaldo institucional y los recursos proporcionados para el desarrollo de esta investigación; a sus tutores, por la orientación académica brindada durante el proceso; y a sus familias, por el apoyo recibido. Asimismo, reconocen la colaboración del Ing. Rodrigo Fernando Morocho Román, cuya orientación técnica desde las etapas iniciales contribuyó significativamente al desarrollo y la puesta en funcionamiento del prototipo. Los autores agradecen a la Universidad Técnica de Machala por el respaldo institucional y los recursos proporcionados para el desarrollo de esta investigación; a sus tutores, por la orientación académica brindada durante el proceso; y a sus familias, por el apoyo recibido. Asimismo, reconocen la colaboración del Ing. Rodrigo Fernando Morocho Román, cuya orientación técnica desde las etapas iniciales contribuyó significativamente al desarrollo y la puesta en funcionamiento del prototipo.

Referencias

Abdellatif, M. M., Desouki, O., & AbdelRaheem, M. (2026). Performance analysis of network automation techniques for dense IP networks. *Scientific Reports*, *16*(1), 9532. https://doi.org/10.1038/s41598-026-40975-9

Adewa, A., Anyah, V., Olufemi, O. D., Oladejo, A. O., & Olaifa, T. (2025). The impact of intent-based networking on network configuration management and security. *Global Journal of Engineering and Technology Advances*, *22*(1), 063-068. https://doi.org/10.30574/gjeta.2025.22.1.0012

Anerousis, N., Chemouil, P., Lazar, A. A., Mihai, N., & Weinstein, S. B. (2021). The origin and evolution of open programmable networks and SDN. *IEEE Communications Surveys & Tutorials*, *23*(3), 1956-1971. https://doi.org/10.1109/COMST.2021.3060582

Esta obra está bajo una licencia *Creative Commons* de tipo (CC-BY-NC-SA).

Grupo Editorial "ALEMA-Pentaciencias" E-mail: alema.pentaciencias@gmail.com

Dsilva, N., Karunakaran, V., Autenrieth, A., Elbers, J.-P., & Bauschert, T. (2025). Network automation using NetDevOps in optical networks. *2025 International Conference on Optical Network Design and Modeling (ONDM)*, 1-3. https://doi.org/10.23919/ONDM65745.2025.11029370

Mazin, A. M., Rahman, R. A., Kassim, M., & Mahmud, A. R. (2021). Performance analysis on network automation interaction with network devices using python. *2021 IEEE 11th IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, 360-366. https://doi.org/10.1109/ISCAIE51753.2021.9431823

Miswar, N., Herman, H., & Riadi, I. (2023). Comparing the performance of ospf and ospf-mpls routing protocol in forwarding tcp and udp packet. *Jurnal Teknik Informatika (Jutif)*, *4*(5), 1237-1247. https://doi.org/10.52436/1.jutif.2023.4.5.1456

Osei-Wusu, F., Asiedu, W., Asamoah, K. F., Muntaka, S. A., Yeboah, D., & Sarfo, E. A. (2025). Automating network programmability and backup on cisco devices using python and netmiko library: A case study of komfo anokye teaching hospital LAN. *Journal of Computing Research and Innovation*, *10*(1), 227-242. https://doi.org/10.24191/jcrinn.v10i1.488

Ren, Y., Wang, Z., Sharma, P. K., Alqahtani, F., Tolba, A., & Wang, J. (2025). Zero trust networks: Evolution and application from concept to practice. *Computers, Materials & Continua*, *82*(2), 1593-1613. https://doi.org/10.32604/cmc.2025.059170

Sassa, A. C., Almeida, I. A. D., Pereira, T. N. F., & Oliveira, M. S. D. (2023). Scrum: A systematic literature review. *International Journal of Advanced Computer Science and Applications*, *14*(4). https://doi.org/10.14569/IJACSA.2023.0140420

Schwaber, K., & Sutherland, J. (2020). *La guía de scrum: La guía definitiva de scrum: Las reglas del juego* (p. 16) [Guía]. Scrum.org y Scrum Inc. https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Spanish-Latin-South-American.pdf

Shah, J. A., & Dubaria, D. (2019). NetDevOps: A new era towards networking & DevOps. *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, 0775-0779. https://doi.org/10.1109/UEMCON47517.2019.8992969

Tao, J., Yuan, R., Liu, Q., & Xia, Q. (2021). Research and implementation of a network based on SDN and multi area OSPF protocol. *2021 IEEE 9th International Conference on Information, Communication and Networks (ICICN)*, 134-138. https://doi.org/10.1109/ICICN52636.2021.9673836

Viuche, S. N., Estupiñán Cuesta, E. P., & Martínez Quintero, J. C. (2025). Comparative performance analysis of software-defined networking vs conventional IP networks using IGP protocols. *Bulletin of Electrical Engineering and Informatics*, *14*(3), 1779-1793. https://doi.org/10.11591/eei.v14i3.9063

Yulán, L., Pazmiño, J. C., & Saavedra, M. (2024). Configuración de una red LAN utilizando python scripting para generar redes definidas por software (SDN). *Revista Científica Élite*, *6*(2), 1-11. https://doi.org/10.69603/itsqmet.vol6.n2.2024.87

Esta obra está bajo una licencia *Creative Commons* de tipo (CC-BY-NC-SA).

Grupo Editorial "ALEMA-Pentaciencias" E-mail: alema.pentaciencias@gmail.com