UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO

CARRERA DE TELECOMUNICACIONES

TEMA:

Diseño e implementación de un entorno virtual de laboratorio para la realización de prácticas de redes LAN en el área de telecomunicaciones.

AUTOR:

Pincay Portilla, Joselin Julexy

Trabajo de titulación previo a la obtención del título de

INGENIERÍA EN TELECOMUNICACIONES

TUTOR:

Ing. Zamora Cedeño, Néstor Armando, Mgs.

Guayaquil, Ecuador

02 de marzo del 2026

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO

CARRERA DE TELECOMUNICACIONES

CERTIFICACIÓN

Certificamos que el presente trabajo de titulación, fue realizado en su totalidad por **Pincay Portilla, Joselin Julexy** como requerimiento para la obtención del título de **Ingeniería en Telecomunicaciones.**

TUTOR:

Néstor Zamora C.

Ing. Zamora Cedeño, Néstor Armando, Mgs.

DIRECTOR DE LA CARRERA

Ing. Bohórquez Escobar, Celso Bayardo, PhD.

Guayaquil, a los 2 días del mes de marzo del año 2026

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO

CARRERA DE TELECOMUNICACIONES

DECLARACIÓN DE RESPONSABILIDAD

Yo, **Pincay Portilla, Joselin Julexy**

DECLARO QUE:

El Trabajo de Titulación, **Diseño e Implementación de un entorno virtual de laboratorio para la realización de prácticas de redes LAN en el área de Telecomunicaciones,** previo a la obtención del título de **Ingeniería en Telecomunicaciones**, ha sido desarrollado respetando derechos intelectuales de terceros conforme las citas que constan en el documento, cuyas fuentes se incorporan en las referencias o bibliografías. Consecuentemente este trabajo es de mi total autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance del Trabajo de Titulación referido.

Guayaquil, a los 2 días del mes de marzo del año 2026

EL AUTOR

Julexy Pincay.

Pincay Portilla, Joselin Julexy

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO

CARRERA DE TELECOMUNICACIONES

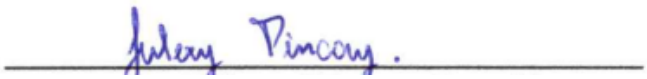
AUTORIZACIÓN

Yo, **Pincay Portilla, Joselin Julexy**

Autorizo a la Universidad Católica de Santiago de Guayaquil a la **publicación** en la biblioteca de la institución del Trabajo de Titulación, **Diseño e Implementación de un entorno virtual de laboratorio para la realización de prácticas de redes LAN en el área de Telecomunicaciones**, cuyo contenido, ideas y criterios son de mi exclusiva responsabilidad y total autoría.

Guayaquil, a los 2 días del mes de marzo del año 2026

EL AUTOR



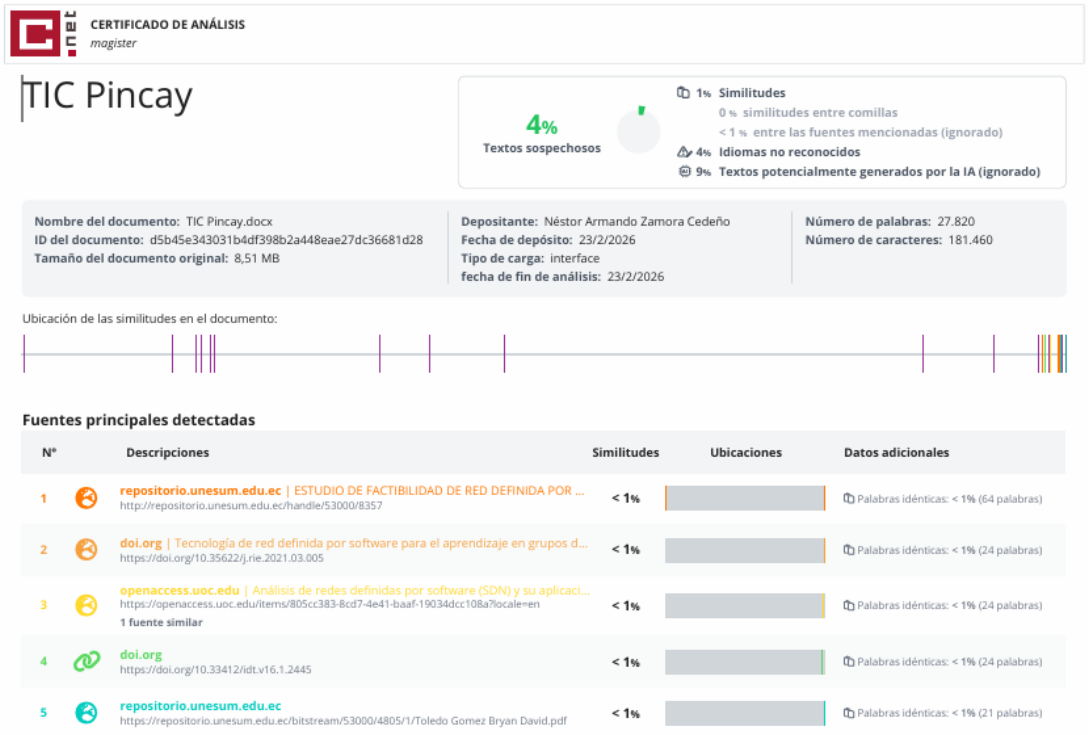
Pincay Portilla, Joselin Julexy

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO

CARRERA DE TELECOMUNICACIONES

INFORME DE COMPILATIO



CERTIFICADO DE ANÁLISIS
magister

TIC Pincay

4%
Textos sospechosos

1% Similitudes
0 % similitudes entre comillas
< 1 % entre las fuentes mencionadas (ignorado)
4% Idiomas no reconocidos
9% Textos potencialmente generados por la IA (ignorado)

Nombre del documento: TIC Pincay.docx
ID del documento: d5b45e343031b4df398b2a448eae27dc36681d28
Tamaño del documento original: 8,51 MB

Depositante: Néstor Armando Zamora Cedeño
Fecha de depósito: 23/2/2026
Tipo de carga: interface
fecha de fin de análisis: 23/2/2026

Número de palabras: 27.820
Número de caracteres: 181.460

Ubicación de las similitudes en el documento:

Fuentes principales detectadas

Nº	Descripciones	Similitudes	Ubicaciones	Datos adicionales
1	repositorio.unesum.edu.ec | ESTUDIO DE FACTIBILIDAD DE RED DEFINIDA POR ... http://repositorio.unesum.edu.ec/handle/53000/8357	< 1%		Palabras idénticas: < 1% (64 palabras)
2	doi.org | Tecnología de red definida por software para el aprendizaje en grupos d... https://doi.org/10.35622/j.rie.2021.03.005	< 1%		Palabras idénticas: < 1% (24 palabras)
3	openaccess.uoc.edu | Análisis de redes definidas por software (SDN) y su aplicaci... https://openaccess.uoc.edu/items/805cc383-8cd7-4e41-baaf-19034dcc108a?locale=en 1 fuente similar	< 1%		Palabras idénticas: < 1% (24 palabras)
4	doi.org https://doi.org/10.33412/idt.v16.1.2445	< 1%		Palabras idénticas: < 1% (24 palabras)
5	repositorio.unesum.edu.ec https://repositorio.unesum.edu.ec/bitstream/53000/4805/1/Toledo Gomez Bryan David.pdf	< 1%		Palabras idénticas: < 1% (21 palabras)

Certifico que después de revisar el documento final del trabajo de titulación **"Diseño e Implementación de un entorno virtual de Laboratorio para la realización de prácticas de redes LAN en el área de Telecomunicaciones"**, presentado por la estudiante **Joselin Julexy Pincay Portilla**, de la carrera de Ingeniería en Telecomunicaciones, donde obtuvo del programa COMPILATIO, el valor de 4**%** de coincidencias, por lo que se aprueba el trabajo para que continúe con el proceso de titulación.

TUTOR:

Néstor Zamora C.

Ing. Zamora Cedeño, Néstor Armando, Mgs.

AGRADECIMIENTOS

A **Dios**, quien me ha dado la dicha y la suerte de estar donde estoy, quien hizo que todo se acomodara para que las cosas pasen justo como deben de pasar, como tener el apoyo de mis padres y liberarme de preocupaciones que hace que muchos jóvenes no puedan llegar o culminar con esta etapa que hoy está terminando para mí.

A mis papás **Jaime y Lorena**, quienes a lo largo de toda mi vida me acompañaron en mi primer día de todo, mi primer día de escuela, mi primer día de colegio, mi primer día de trabajo y mi primer día de universidad, porque sí, siempre tuvieron razón cuando me decían "Todo lo que llegues a ser algún día, también es porque nosotros estuvimos detrás" y todo lo que soy hoy, también lo debo a las acciones de ellos. Gracias por ayudarme a crear buenos recuerdos, a forjarme como persona y por enseñarme que "Un lápiz, pesa menos que una pala"

A mis hermanos; **Madeline** por ser mi más grande ejemplo de que todo con perseverancia se puede lograr, **Jordana** por mostrarme que la madurez no depende de tu edad, **Steven** por enseñarme que la vida no tiene que ser siempre seria, siempre hay algo que disfrutar y **Daniela** quien me recordó lo que era imaginar, porque cada vez que llegaba de la U, narraba mi vida como un capitulo de una novela a la que llamó "Las aventuras de Juli, la hormiga"

Y a todas las personas que estuvieron a lo largo de toda mi vida hasta llegar a este proceso, desde mi abuelita **Jacinta,** quien me enseñó a leer, hasta **Ninibeth**, **Angel**, y todos mis amigos de la U quienes, con sus ocurrencias, hacían bonitos mis días ahí.

DEDICATORIA

Para mi papá, quien en vida fue **Jaime Rodolfo Pincay López**, esta dedicatoria quedará guardada siempre, igual que todo el amor que sentías por mi y yo por ti. Sigue guiando mi camino desde el cielo, que yo aquí en la tierra seguiré honrando tu memoria.

Para mi mamá, **Silvia Lorena Portilla Navarro**, quien gracias a Dios sigue acompañándome en todos mis caminos y tiene la fortaleza para seguir enseñándome las cosas buenas y malas que aún tengo que aprender desde su experiencia.

La buena crianza que me dieron se ve reflejada cuando alguien reconoce una buena acción que llego a realizar. Eso habla bien de mí, pero mucho más de ustedes.

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO

CARRERA DE TELECOMUNICACIONES

TRIBUNAL DE SUSTENTACIÓN

f. _____

Ing. Celso Bayardo Bohórquez Escobar, PhD.

DIRECTOR DE CARRERA

f. _____

Ing. Ricardo Xavier Ubilla Gonzalez, MsC.

COORDINADOR DEL ÁREA

f. _____

Ing. Ronnie Alexander Bonilla Sanchez, MsC.

OPONENTE

INDICE GENERAL

INDICE DE FIGURAS

INDICE DE TABLAS

RESUMEN

Esta tesis aborda el desafío de la brecha tecnológica y el costo de crear laboratorios de redes de datos físicos en la facultad de educación técnica para el desarrollo (FETD). El objetivo principal fue diseñar e implementar un entorno de laboratorio virtualizado, que opera bajo el paradigma de redes definidas por software (SDN), con el emulador Mininet y el controlador Ryu en el estándar OpenFlow 1.3.

Este enfoque implicó diseñar dos topologías fundamentales (Estrella y árbol jerárquico) y utilizar scripts de Python para automatizarlas, optimizando así el despliegue de una estación de trabajo piloto con recursos de hardware intermedios (Intel Core i5-5200U). Las pruebas de conectividad y rendimiento con herramientas, incluidas Iperf y Wireshark, mostraron una tasa de transferencia (orden de Gbits/segundo) y una latencia de procesamiento reactivo de menos de 1 ms: el uso de CPU fue inferior al 60% por lo que el sistema se mantuvo estable.

Finalmente, el laboratorio se ensambló en formato OVF (.OVA) que podría usarse rápidamente para desplegarlo en cualquier ubicación del campus. Se concluye que la solución propuesta no solo elimina las necesidades hardware propietario y costoso, sino que establece una plataforma pedagógica flexible y escalable que se ajusta a las tendencias actuales en infraestructura como código (IaC) y refuerza el proceso de enseñanza-aprendizaje en la carrera de telecomunicaciones.

Palabras clave: *Redes, SDN, Mininet, Ryu, Virtualización, Laboratorio, Topologías, Protocolos.*

ABSTRACT

This thesis addresses the challenge of the technological gap and the cost of creating physical data network laboratories in the faculty of technical education for development (FETD). The main objective was to design and implement a laboratory environment virtualized networking, which operates under the software-defined networking (SDN) paradigm, with the Mininet emulator and the Ryu controller in the OpenFlow 1.3 standard.

This approach involved designing two fundamental topologies (Star and Hierarchical Tree) and use Python scripts to automate them, thus optimizing the deployment of a pilot workstation with intermediate hardware resources (Intel Core i5-5200U). Connectivity and performance testing with tools, including Iperf and Wireshark showed a transfer rate (order of Gbits/second) and a reactive processing latency of less than 1 ms: CPU usage was less than 60% so the system remained stable.

Finally, the laboratory was assembled in OVF format (. OVA) that could be quickly used to deploy it to any location on campus. It is concluded that the proposed solution not only eliminates the needs of proprietary and expensive hardware, rather, it establishes a flexible and scalable pedagogical platform that adjusts to current trends in infrastructure as code (IaC) and reinforces the teaching-learning process in the telecommunications career.

Keywords: *Networks, SDN, Mininet, Ryu, Virtualization, Lab, Topologies, Protocols.*

CAPITULO 1

DESCRIPCION GENERAL DEL TRABAJO DE TITULACION

1.1 INTRODUCCIÓN

La educación en el área de telecomunicaciones ha tenido un desarrollo notable, requiriendo cada vez más la inclusión de herramientas prácticas que posibiliten a los alumnos interactuar con escenarios reales. En el contexto de las redes de datos, entender la teoría de los protocolos no es suficiente si esto no va acompañado de la configuración y diagnóstico de dispositivos activos.

Sin embargo, la infraestructura física necesaria para poder replicar redes complejas y dispositivos de red, como switches, routers y cableados estructurados, implica un alto costo y limita la escalabilidad de los laboratorios en los entornos universitarios. El acceso a hardware frecuentemente es limitado en comparación con el número de estudiantes y crear configuraciones de topologías complejas o de gran escala suele ser poco práctico, ya que, existen restricciones físicas y costos de mantenimiento del equipo. Todas estas limitaciones pueden acortar el alcance de las prácticas experimentales, afectando así el desarrollo de habilidades técnicas.

Frente a esta problemática, es cuando surge la virtualización de redes y se presenta como una alternativa tecnológica efectiva, abordando los obstáculos físicos sin comprometer la exactitud técnica.

Herramientas como Mininet permiten emular el comportamiento de redes completas usando el mismo núcleo o kernel del sistema operativo, ofreciendo un entorno flexible donde se puede experimentar con arquitecturas de red tradicionales y redes definidas por software, sin la necesidad de grandes inversiones en hardware adicional.

Por ende, el presente trabajo de titulación propone el diseño e implementación de un entorno de laboratorio virtual basado en Mininet para la carrera de Telecomunicaciones de la UCSG. El proyecto tiene como objetivo crear un conjunto de escenarios y guías prácticas que capaciten a los estudiantes para simular y examinar redes LAN con un nivel de realismo y escalabilidad que supera las capacidades de la infraestructura física existente. La investigación busca comprobar que este entorno

virtual representa una herramienta didáctica eficaz y técnica para el análisis del tráfico y la configuración de protocolos.

1.2 ANTECEDENTES

La incorporación de entornos virtuales en la educación de futuros ingenieros en telecomunicaciones ha sido investigada en varias instituciones de enseñanza superior en Ecuador, tratando de ofrecer respuestas a la creciente necesidad de prácticas especializadas.

A nivel nacional, (Paucar Asqui, 2022) en su proyecto de graduación presentado en la Universidad Nacional de Chimborazo (UNACH), analizó el "Análisis e implementación de un prototipo de red SDN". En esta investigación, los autores diseñaron una arquitectura de red utilizando máquinas virtuales sobre VirtualBox con el sistema operativo Ubuntu, donde integraron la herramienta Mininet. El enfoque del estudio fue examinar el funcionamiento de controladores de código abierto (como OpenDaylight y POX) y estudiar el tráfico con Wireshark. Esta investigación contribuyó con un hito técnico importante: estableció que la simulación de infraestructuras de red altamente complicadas y el análisis de paquetes en tiempo real utilizando Mininet sin dispositivos físicos puede realizarse con la virtualización combinada con Mininet,.

Por otro lado, centrado en mejorar la accesibilidad para los estudiantes (Estévez Almeida, 2022) de la Universidad Técnica del Norte (UTN), desarrolló una “Infraestructura como servicio para desarrollo de prácticas SDN y NFV”. Su análisis de la situación, demostró que los estudiantes enfrentaban dificultades para realizar prácticas de laboratorios debido a las altas exigencias de hardware, que requieren los simuladores tradicionales. Como respuesta a esta problemática, implementaron un sistema de instancias virtuales, que permitió a los alumnos poder acceder a herramientas y servicios ya disponibles. Los resultados de este proyecto, concluyeron que ofrecer entornos virtuales preconfigurados, mejora significativamente la experiencia en el ámbito de aprendizaje, eliminado barreras técnicas y de procesamiento en los equipos personales de los alumnos.

Aún así, a pesar de estos avances en otras universidades del país, en la carrera de Telecomunicaciones de la UCSG aún no se ha establecido un uso estandarizado de una plataforma de emulación basada en Mininet, para de esta manera poder realizar

prácticas de redes LAN. Esta tesis pretende solventar ese vacío, ajustando estas tecnologías a las necesidades curriculares y la infraestructura actual del laboratorio de telecomunicaciones. Aunque en estudios anteriores se ha demostrado la eficacia técnica de Mininet y la virtualización en el país, sus métodos se enfocaron en el análisis comparativo de controladores SDN sofisticados o implementaciones complejas en la nube. A diferencia de esos enfoques, esta investigación se caracteriza por su énfasis pedagógico y operativo centrado en redes LAN tradicionales, con el objetivo de abordar la falta de infraestructura física en el laboratorio de la UCSG mediante el desarrollo de prácticas guiadas que no necesitan de conexión a servidores externos ni de configuraciones complicadas de controladores.

1.3 PLANTEAMIENTO DEL PROBLEMA

La educación profesional en el ámbito de Telecomunicaciones de la UCSG demanda de manera esencial la combinación de saberes teóricos y vivencias prácticas. Sin embargo, existe una falta de equipo físico, como conmutadores, enrutadores y controladores específicos, dedicados a la instrucción práctica sobre redes de datos. A pesar de que la universidad dispone de laboratorios informáticos comunes, no se cuenta con un laboratorio especializado en redes que cuente con dispositivos de red de propósito específico, donde los alumnos puedan realizar configuraciones, hacer cableado o implementar topologías.

Esta ausencia de infraestructura física impide que se desarrolle un camino educativo, más allá de lo que se aprende como teoría o a través de simuladores básicos que no son estandarizados. Al no existir dispositivos para manipular, los estudiantes no pueden experimentar fenómenos reales de red como la congestión, la latencia o la segmentación del tráfico en un entorno controlado.

Como resultado inmediato, los futuros ingenieros enfrentan una desconexión seria, entre los conceptos teóricos y la práctica efectiva. Se gradúan con una comprensión teórica de los protocolos de red de área local, pero sin haber llevado a cabo la configuración de una red operativa, lo que les genera una desventaja competitiva significativa al intentar ingresar en el mercado laboral.

1.4 JUSTIFICACIÓN

Dado el desafío de acceder a dispositivos físicos de red, se creará un ambiente operativo para realizar ejercicios de redes. Mininet transformará los ordenadores

existentes en verdaderos nodos de red, compensando la ausencia de hardware con una solución de emulación altamente precisa que emplea el núcleo de Linux para replicar el comportamiento de los dispositivos físicos, incluso permitiendo el uso de herramientas reales para el análisis de tráfico como Wireshark.

Asimismo, es importante considerar que equipar el laboratorio desde cero con hardware especializado para los estudiantes implicaría una enorme inversión económica. Esta tesis sugiere una alternativa inmediata que no conlleva gastos, completamente basada en software de código abierto y gratis, utilizando las estaciones de trabajo ya disponibles y eludiendo costos relacionados con la compra de licencias o equipos.

De este modo, se transforma una enseñanza que era meramente teórica en un enfoque práctico y activo, lo que permite a los estudiantes confirmar sus conocimientos y adquirir las habilidades prácticas que se requieren al finalizar su formación.

1.5 OBJETIVOS

1.5.1 Objetivo general

- Diseñar e implementar un entorno virtual de laboratorio basado en el emulador de redes Mininet para la realización de prácticas de redes LAN, optimizando los recursos tecnológicos y fortaleciendo el aprendizaje práctico en la carrera de Telecomunicaciones de la UCSG.

1.5.2 Objetivos específicos

- Analizar los requerimientos técnicos del software de emulación Mininet para el diseño de entornos virtuales, verificando la compatibilidad con las computadoras del laboratorio.
- Diseñar las topologías de red virtuales “Estrella y Árbol” definiendo los esquemas de direccionamiento IP y los protocolos de comunicación como ARP e ICMP, que los estudiantes configurarán en la plataforma de emulación Mininet.
- Implementar el entorno virtual en una estación de trabajo piloto, realizando pruebas de verificación de conexión y análisis de tráfico con herramientas de monitoreo como Wireshark, dentro de Mininet, para asegurar la funcionalidad y estabilidad del sistema.

- Realizar el manual de implementación y las guías de laboratorio paso a paso con la herramienta Mininet.

1.6 HIPÓTESIS

Con la implementación de un laboratorio virtual utilizando la herramienta de emulación Mininet, se suple la carencia y la falta de los equipos físicos para realizar prácticas en la FETD, específicamente en la carrera de Telecomunicaciones de la UCSG, obteniendo como resultados que los estudiantes puedan tener un entorno funcional para validar sus conocimientos teóricos.

1.7 METODOLOGÍA DE LA INVESTIGACIÓN

1.7.1 Tipo de investigación

El estudio será de tipo aplicado y experimental, ya que se busca solucionar la falta de equipos usando los conocimientos teóricos aprendidos dentro de la carrera de Telecomunicaciones. Además, se realizarán pruebas en el software Mininet para verificar la correcta funcionalidad, entregando un proyecto factible para la realización de prácticas.

1.7.2 Esquema metodológico

Para llevar a cabo el proyecto y cumplir con los objetivos anteriormente dispuestos, el trabajo se estructurará bajo el siguiente esquema que contiene 3 fases secuenciales:

Fase 1 – Análisis y diagnóstico.

Recopilación de información sobre el hardware disponible dentro de las PCs del laboratorio de telecomunicaciones de la UCSG y selección de la versión del sistema operativo (LINUX/UBUNTU) y del emulador Mininet que se ajusten a dichos recursos.

Fase 2 – Diseño de la solución.

Diseño de las topologías de red lógicas (como Estrella o Árbol) y se definirá el direccionamiento de IP necesario; para posterior a ello realizar la redacción de la guía paso a paso, que servirá como material de apoyo docente.

Fase 3 – Implementación.

Aquí se realizará la instalación y configuración del software en los equipos, así mismo, la carga de los scripts de topología en una estación de trabajo piloto.

Posteriormente, se verificará que esté funcionando de manera efectiva con pruebas de ping y análisis de tráfico, asegurando que la solución es estable y funcional.

CAPITULO 2 MARCO TEÓRICO

2.1 FUNDAMENTOS DE REDES DE DATOS

Para comprender el alcance del entorno virtual propuesto, es esencial establecer los conceptos fundamentales de las redes de datos.

2.1.1 Redes de área local (LAN)

En el ámbito de las telecomunicaciones, existe un consenso claro con respecto a la definición de las redes LAN. Según coinciden autores fundamentales como (Tanenbaum & Wetherall, 2021) y (Kurose & Ross, 2021) una red de área local (LAN), está definida como una infraestructura de comunicación privada que se caracteriza por conectar sistemas finales, operando dentro de un área geográfica limitada, diseñadas para compartir recursos e intercambiar información con tiempos de retardo bajos y altas velocidades de transmisión. Comúnmente operan bajo estándares Ethernet que permiten la administración local del tráfico y la seguridad y se distingue de otros tipos de redes por tres factores clave: su tamaño restringido, su tecnología de transmisión y su topología.

LAN vs WAN

Las redes LAN se caracterizan por sus altas tasas de transmisión de datos, a diferencia de las redes de área amplia, mejor conocidas como redes WAN (por sus siglas en inglés Wide Area Networks); las cuales se extienden sobre una gran área geográfica, que a menudo abarca un país o continente.

Otra característica importante es que las redes LAN dependen de un hardware propio, por el contrario de las redes WAN, que utilizan circuitos de telecomunicaciones que los proporcionan empresas de servicios comunes para la transmisión de datos a largas distancias.

Tabla 1

Comparación Técnica entre Redes LAN y Redes WAN

Característica	Red LAN (Laboratorio Virtual)	Red WAN
Control de infraestructura	Total, y centralizado, lo que lo hace ideal en SDN.	Compartido con proveedores (ISP).
Enfoque académico	Switching, ARP, VLANs e IP local.	Enrutamiento complejo (BGP, MPLS).

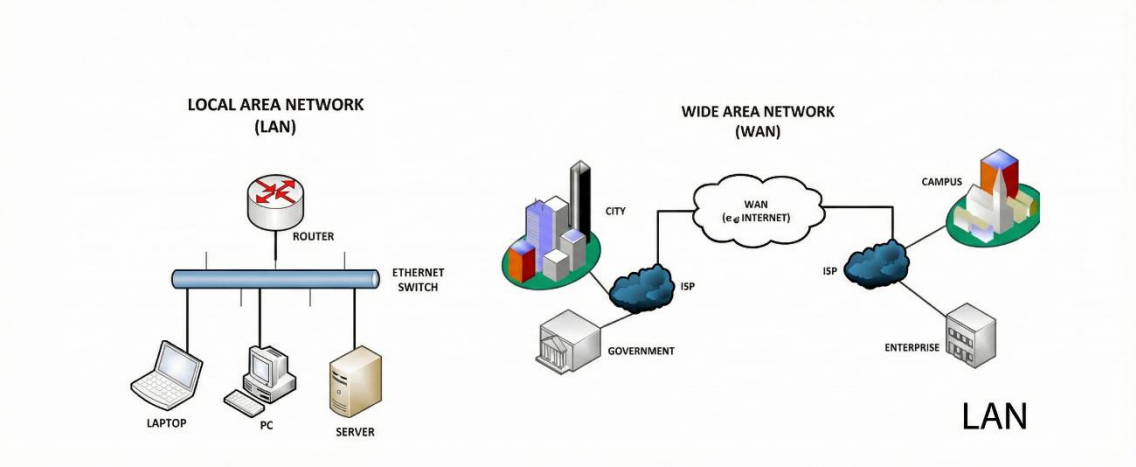
Tasa de errores	Muy baja	Más elevada debido a la distancia y medios externos.
Recursos de hardware	Optimizado para PCs de laboratorio.	Requiere mayor capacidad de cómputo.

Nota. Comparación de los atributos clave que diferencian el entorno de pruebas propuesto frente a redes externas. *Fuente:* Autor.

Su importancia en este estudio, se debe a que se necesita comprender los límites del laboratorio propuesto. En el contexto de este laboratorio virtual, todas las emulaciones se van a realizar bajo la arquitectura de una LAN ethernet, que es el estándar predominante en la industria actual; aunque la herramienta a utilizar (Mininet) si tiene la capacidad técnica de emular enlaces WAN mediante la simulación de retardos y pérdidas de paquetes, el enfoque principal de las prácticas diseñadas será el entorno local de alta velocidad.

Figura 1

Comparación de alcance geográfico entre una red LAN y una red WAN



Nota. Comparativa de arquitecturas de red: una LAN para conectividad local de dispositivos y una WAN para la interconexión de nodos a gran escala geográfica. *Fuente*: (Microinstalaciones, 2021)

2.1.2 Topologías de red

En el diseño de redes modernas, la topología de red se refiere a la estructura física y lógica en la que los nodos se interconectan y comunican. La arquitectura física de una red local no solo determina la disposición de los cables, sino que también la eficiencia del intercambio de datos y la robustez del sistema ante fallos. Según la

literatura más reciente, la topología elegida impacta directamente en la latencia, el rendimiento (throughput) y también la capacidad de gestión del controlador en entornos definidos por software (SDN).

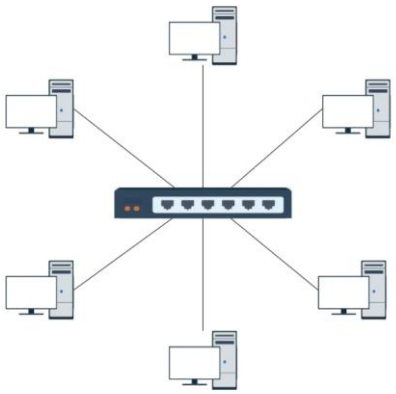
El diseño de redes LAN se ha estandarizado hacia estructuras que facilitan la administración inteligente del tráfico. De acuerdo con (Forouzan, 2021) la topología define la relación de los enlaces y los dispositivos que los interconectan. Para los escenarios de este laboratorio virtual, se han seleccionado dos arquitecturas fundamentales que nativamente son soportadas por el emulador Mininet, las cuales, a su vez, son de las arquitecturas más relevantes en la infraestructura moderna para redes locales y de acceso, representando el estándar en ámbitos corporativos y de campus:

A. Topología en Estrella

La topología en estrella, conocida en inglés como Star Topology, es la configuración que predomina en las redes de área local (LAN) actuales. En este modelo, todos los hosts (nodos finales) se conectan a un dispositivo central de conmutación. Lo que significa que, a diferencia de las topologías compartidas del pasado, la topología estrella moderna utiliza conmutación de paquetes para evitar colisiones, así lo describen (Kurose & Ross, 2021) en donde describen esta topología como, HUB AND SPOKE (centro y radios), donde el conmutador central actúa como único intermediario ante el tráfico, eliminando las colisiones de datos típicas de topologías antiguas como el bus.

Figura 2

Topología de red en estrella con conmutador central



Nota. Esquema de conexión donde los nodos finales se enlazan individualmente a un dispositivo central para evitar colisiones. *Fuente:* (ProcessOn, 2025)

Rendimiento en Mininet

En los estudios más recientes sobre el rendimiento de controladores SDN, se ha demostrado que la topología en estrella es la que ofrece la menor fluctuación (jitter) y latencia, frente a escenarios de tráfico moderado. En la investigación de (Munusamy & Shukla, 2024) donde se evaluaron controladores como RYU y ONOS en Mininet, concluyó que la topología en estrella es ideal para los escenarios que requieren alta velocidad de respuesta y una simple gestión de flujos.

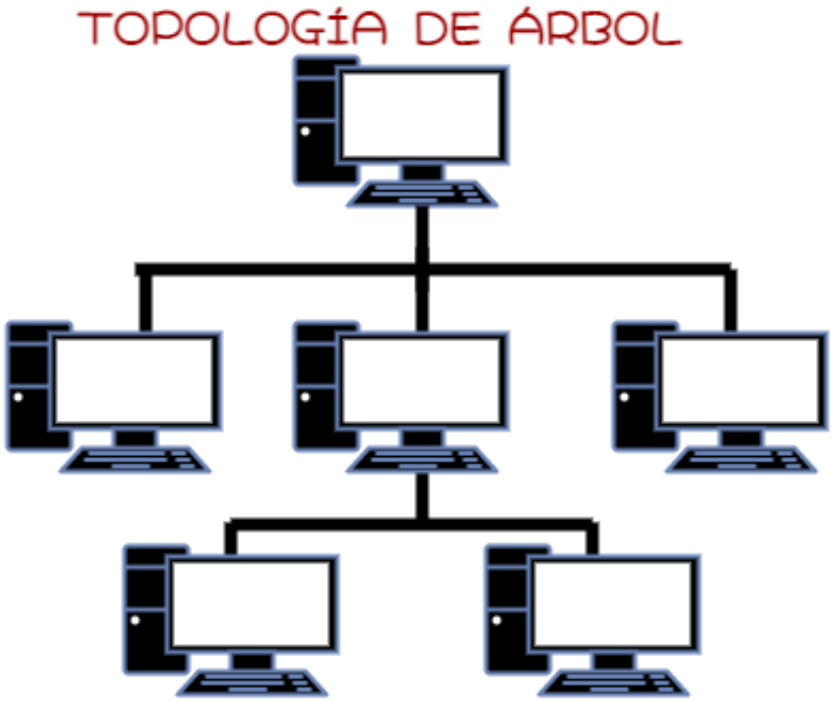
B. Topología en Árbol

La topología en árbol, conocida en inglés como Tree Topology, es una estructura jerárquica que extiende la topología en estrella, se encarga de organizar todos los dispositivos en diferentes niveles (Acceso, Distribución y Núcleo). Este es el estándar más utilizado en redes universitarias y empresariales escalables. (Forouzan, 2021) indica que esta estructura permite que la red se segmente, facilitando así la administración y reduciendo el dominio de difusión (broadcast domain) cuando se implementan VLANs.

Ya se han realizado pruebas en Mininet con esta topología, según lo revisado por (Keerthana et al., 2022) se observó que la topología en árbol es crítica para evaluar el rendimiento de los algoritmos de enrutamiento del controlador, obligando a los paquetes a atravesar múltiples switches (saltos) para poder llegar a su destino. De esta manera, permite la medición de métricas reales en el retardo de ida y vuelta (RTT) que no son visibles en redes planas.

Figura 3

Topología jerárquica o en árbol



Nota. Estructura similar a las ramas de un árbol, que mantiene un nodo raíz que conecta varios dispositivos secundarios, interconectando múltiples redes en estrella. *Fuente*: (Blanco Torres, 2024)

2.2 MODELOS DE REFERENCIA Y PROTOCOLOS

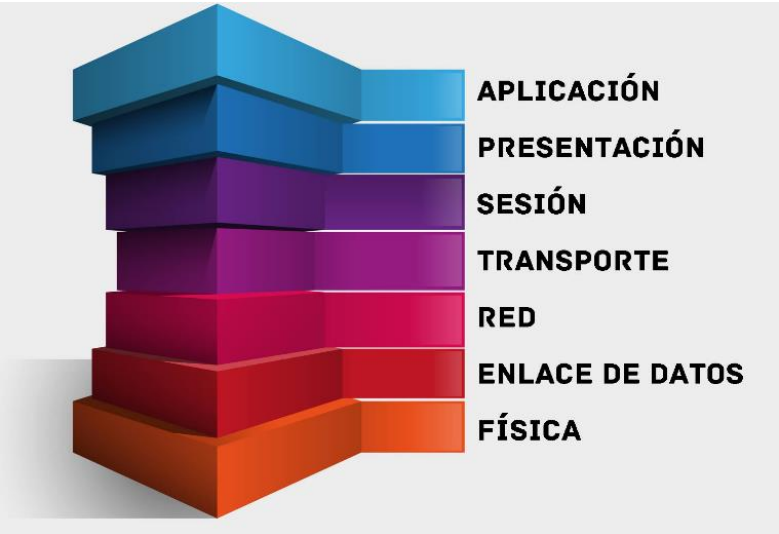
Para el análisis del flujo o tránsito de datos en una red, es esencial consultar los modelos que normalizan la comunicación. En la actualidad, en el ámbito de la ingeniería de redes y telecomunicaciones, se destacan dos estructuras principales: el modelo teórico OSI, conocido así por su nombre en inglés: Open Systems Interconnection (Interconexión de Sistemas Abiertos) y el modelo práctico TCP/IP.

2.2.1 Modelo de interconexión de sistemas abiertos (OSI)

El modelo OSI es un estándar teórico de referencia desarrollado por la ISO, que divide las funciones de una red en siete capas diseñadas para que distintos sistemas puedan interactuar, con el fin de garantizar la interoperabilidad. Según (Forouzan, 2021) este marco garantiza la interoperabilidad entre sistemas heterogéneos mediante una jerarquía donde cada nivel va prestando servicios al superior.

Figura 4

Estructura jerárquica de las siete capas del modelo OSI



Nota. Ilustración de la estructura jerárquica de las siete capas definidas del modelo OSI. Fuente: (González, 2021)

Aunque las redes SDN centralizan el control de la red, el plano de datos sigue respetando la estructura de encapsulamiento del modelo OSI; su importancia en este proyecto radica en la estructura de los encabezados de los paquetes. Para fines de este proyecto, se prioriza el estudio de la capa de Enlace de datos (Capa 2) y la capa de Red (Capa 3):

Capa de enlace de datos (Capa 2)

Es la responsable de la transferencia fiable de información a través de un circuito de transmisión de datos físico. De acuerdo con (Tanenbaum & Wetherall, 2021) en esta capa operan los conmutadores, los cuales son los switches; también se utiliza el direccionamiento físico o MAC Address, para identificar los dispositivos en la misma red local y la detección de errores en los mismos. Esta capa es fundamental en el entorno de las redes SDN, pues el protocolo OpenFlow gestiona el reenvío basado en los puertos del switch.

Capa de red (Capa 3)

Según (Kurose & Ross, 2021), es la que se encarga del enrutamiento de paquetes a través de la red, aquí operan los routers y se utiliza el direccionamiento lógico (IP) para conectar redes distintas. Esta capa es esencial, porque aquí se ejecuta

el protocolo ICMP, el cual será utilizado en esta tesis para validar la conectividad mediante el comando ping.

2.2.2 Modelo TCP/IP

El modelo TCP/IP no es una referencia teórica igual que el modelo OSI, este modelo es la arquitectura práctica diseñada para la implementación real en redes, sobre la que funciona internet y la mayoría de las redes modernas, incluyendo a las redes virtuales generadas por Mininet. Este modelo condensa las siete capas OSI en cuatro capas funcionales:

- **Capa de Acceso a la Red:** Combina 2 capas del modelo OSI, la capa física y la de enlace.
- **Capa de Internet:** Equivale a la capa de red (IP) del modelo OSI.
- **Capa de Transporte:** Gestiona la comunicación de extremo a extremo mediante el protocolo orientado a la conexión (TCP) y no orientado a la conexión (UDP).
- **Capa de Aplicación:** Agrupa las tres capas superiores del modelo OSI (Sesión, Presentación y Aplicación). En esta capa residen los servicios de red visibles para el usuario.

Este modelo es la base de la pila de protocolos utilizada por el sistema operativo Linux, el mismo sobre el que se despliega la herramienta de simulación Mininet. En telecomunicaciones existe una estructura funcional de los sistemas de red, que comúnmente se asocian a las capas de los modelos utilizados en esta investigación y a la arquitectura de gestión de redes modernas (SDN). Las capas del modelo, se separan en tres diferentes planos, separando la interacción del usuario (aplicación), la lógica del enrutamiento (control) y el transporte físico de la información (datos). En el desarrollo del laboratorio virtual, es necesario distinguir entre las capas del modelo de comunicación TCP/IP y los planos funcionales de la arquitectura SDN, ya que ambos conceptos se integran entre sí.

2.2.3 Protocolos de comunicación en redes de datos

Para garantizar la validez de las pruebas de conectividad y rendimiento de la herramienta a utilizar (Mininet), es necesario definir los protocolos que rigen el intercambio de información en la capa de red y enlace; para que el laboratorio virtual funcione, se analizarán los protocolos descritos a continuación:

A. Protocolo IP (Internet Protocol)

Este protocolo es el pilar fundamental de la capa de internet en el modelo TCP/IP y su función primordial es el transportar los datagramas desde el origen hasta el destino, definiendo así el esquema de direccionamiento que identifica a cada host en la red virtual, a tráves de una infraestructura de red conmutada. Según (Peterson & Davie, 2021) este protocolo está diseñado bajo el principio de interoperabilidad, de esta manera actúa como un protocolo común que permite la comunicación entre tecnologías de red heterogéneas.

B. Protocolo ARP (Address Resolution Protocol)

Es el protocolo de resolución de direcciones y su función es fundamental en redes LAN, ya que permite asociar una dirección IP conocida con una dirección física MAC desconocida, es decir que, mientras los usuarios y aplicaciones si entienden la dirección IP, el hardware de red (tarjeta de red y switches) se comunican mediante la MAC. Según lo que explican (Tanenbaum & Wetherall, 2021) el protocolo ARP actúa como un puente traductor, enviando una petición de difusión (broadcast) a la red preguntando ¿Quién tiene la IPx? Para obtener la MAC que le corresponde.

C. Protocolo ICMP (Internet Control Message Protocol)

El protocolo de mensajes de control de internet es un auxiliar del nivel de red, que lo utilizan los dispositivos para reportar errores y realizar diagnósticos operativos. No se utiliza para enviar datos de usuario, sino para verificar si un camino está disponible, la herramienta más utilizada por ICMP es el “ping”, según lo que detallan (Kurose & Ross, 2021) esta y otras herramientas comunes funcionan enviando mensajes Echo Request y esperando un Echo Reply.

2.3 EL PARADIGMA DE LAS REDES TRADICIONALES

Para comprender la transformación que proponen las redes definidas por software (SDN), es importante que se analice la arquitectura que ha regido las telecomunicaciones en los últimos años. Según (Forouzan, 2021), tradicionalmente, una red de datos se define como un conjunto de dispositivos tecnológicos (nodos) interconectados entre sí mediante medios, ya sean físicos o inalámbricos. La administración de estas infraestructuras ha tenido evolución desde sistemas asilados hacia un modelo descentralizado y estático, donde todos los dispositivos poseen su

propia lógica de control autónoma, es por lo que hoy se identifica como el paradigma tradicional.

La estructura de las redes convencionales se basa en un modelo de gestión sin centralización, donde cada nodo alberga la inteligencia de manera individual. Según lo indicado por (Santamaría Sandoval, 2020) la configuración clásica se distingue por la utilización de dispositivos autónomos, cuya configuración y gestión necesitan intervención manual directa, lo que genera significativas restricciones en agilidad operativa para satisfacer las exigencias de los actuales modelos de telecomunicaciones.

Las redes tradicionales establecen un modelo arquitectónico vertical, en este esquema cada dispositivo de red (como switches o routers) combinan dos funciones importantes en su hardware:

- **Plano de control:** Se refiere a la "inteligencia" del dispositivo y es el que decide el camino que deben seguir los paquetes, a través de procesos de toma de decisiones basadas en tablas de enrutamiento y conmutación.
- **Plano de datos:** También llamado plano de reenvío, es el elemento responsable de trasladar físicamente los paquetes de una interfaz de entrada a otra de salida, conforme a las instrucciones dictadas por el plano de control.

Este es uno de los aspectos técnicos más importantes del modelo, ya que describe la unión de sus funciones internas. (Vega Guallpa et al., 2022) menciona que las redes tradicionales se consideran sistemas en los que el plano de control y el plano de datos están verticalmente integrados en el mismo dispositivo, ya sean routers o switches. Esta arquitectura cerrada significa que cada dispositivo opera como una entidad autónoma que debe manejar protocolos de enrutamiento y conmutación de forma independiente.

Por lo tanto, esta estrategia enfrenta importantes retos en entornos de alta demanda, según lo que argumenta (Zúñiga Sánchez et al., 2025) indicando que la operación independiente de cada nodo limita la flexibilidad y escalabilidad de la infraestructura, obstaculizando una gestión centralizada y adaptativa.

De hecho, la inflexibilidad de las redes tradicionales, al no contar con una separación funcional entre planos, restringe la capacidad de la red para adaptarse a cambios en el tráfico o nuevos requerimientos de servicios, lo que señala el momento clave para la evolución hacia redes definidas por software (SDN).

2.3.1 Limitaciones críticas: Rigidez y falta de programabilidad

A medida que la cantidad de datos se incrementa de manera considerable, a causa de la virtualización, el modelo tradicional enfrenta obstáculos que impulsan la búsqueda de nuevas estructuras arquitectónicas. De acuerdo con (Ariganello, 2020) las arquitecturas de red convencionales tienen una principal causa de sus limitaciones y esto se debe a la gestión descentralizada que manejan, en los siguientes puntos se detalla cómo se presenta esto:

- **Administración Salto a Salto (Hop-by-hop):** La configuración de reglas de red o seguridad se lleva a cabo de manera aislada en cada unidad mediante interfaces de líneas de comandos (CLI). Este enfoque manual es altamente propenso a errores de las personas y complica la escalabilidad en grandes entornos corporativos.
- **Dependencia del fabricante (Vendor Lock-in):** El software de control está altamente conectado al hardware, es decir que, si un ingeniero necesita alguna nueva característica, funcionalidad o protocolo, normalmente debe esperar a qye el proveedor lance una actualización oficial o en muchos casos, se debe cambiar todo el equipo físico.
- **Estatismo operativo:** Las redes tradicionales son rígidas, carecen de flexibilidad y no se adaptan rápidamente a las variaciones dinámicas de los centros de datos actuales. Cualquier modificación en la estructura necesita una acción manual que puede extenderse a horas o días.

2.3.2 Virtualización de redes

Precisamente por las limitaciones de los sistemas aislados correspondientes a las redes tradicionales, es donde surge la virtualización de redes, como una capa de abstracción que desacopla los servicios de la red del hardware físico específico. Según lo que explica (Vega Guallpa et al., 2022) la virtualización permite que se creen

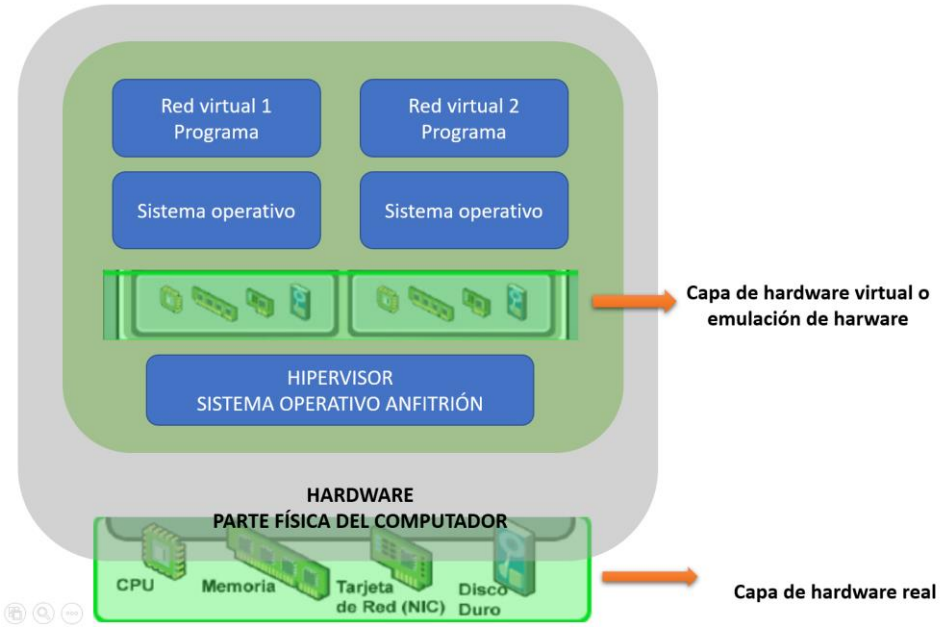
múltiples redes virtuales sobre una única infraestructura física, optimizando el uso de recursos y permitiendo que su segmentación sea más eficiente.

En este marco, la virtualización se establece como la tecnología clave para la implementación de redes SDN, facilitando la gestión eficiente de los recursos de telecomunicaciones; en contraste con las redes tradicionales, donde cada dispositivo funciona como una unidad física y lógica que no se puede dividir, la virtualización posibilita la formación de diversas redes lógicas independientes.

Este principio es fundamental para avanzar en la flexibilidad operativa que se encuentra ausente en diseños tradicionales, ya que proporciona la funcionalidad de programar y automatizar servicios sin estar limitados por una topología física rígida. Como sostienen (Zúñiga Sánchez et al., 2025) esta innovación no solo promueve el aprendizaje en entornos estructurados, sino que representa un elemento fundamental para el desarrollo de SDN al permitir que la gestión de recursos abstractos se realice de forma central y dinámica.

Figura 5

Arquitectura de capas de virtualización de hardware y red



Nota. La virtualización posibilita la existencia de múltiples topologías lógicas utilizando el mismo hardware. Adaptado de *¿Qué es la virtualización?,* por Un poco de Java. Fuente: Autor.

2.3.3 Hipervisores

El elemento fundamental que hace viable la virtualización es el "hipervisor" o también conocido como "monitor de máquina virtual". Al diseñar este laboratorio, es importante distinguir entre las dos clases de hipervisores según su configuración:

- **Hipervisor tipo 1 (Nativo o Bare-metal):** Este tipo opera directamente sobre el hardware físico, gestionando los recursos sin depender de un sistema operativo como anfitrión. Aunque ofrecen un alto rendimiento, su instalación en laboratorios académicos puede resultar complicada, ya que requiere un hardware específico y el formateo de las estaciones de trabajo.
- **Hipervisor tipo 2 (Hospedado):** Este se ejecuta como un programa sobre un sistema operativo tradicional, esta arquitectura es adecuada para ambientes académicos. ya que permite la implementación de máquinas virtuales sin modificar la configuración fundamental de los dispositivos.

Para este proyecto se optará por el hipervisor Oracle VirtualBox (tipo 2), porque permitirá a los estudiantes la implementación del entorno Mininet en los equipos del laboratorio de telecomunicaciones de la facultad técnica, todo con el fin de que se ejecute el programa de manera no intrusiva.

2.3.4 Virtualización de las funciones de red (NFV)

Estrechamente vinculado a SDN, es donde nace el concepto de virtualización de funciones de red, de acuerdo con (González et al., 2020) esta técnica permite que funciones que solían requerir hardware especializado, se ejecuten dentro del software de máquinas virtuales.

La colaboración entre SDN y NFV hace que la red sea configurable no solo en su transporte (SDN), sino también en los servicios (NFV). Según (Jácome Paredes, 2025) la implementación de virtualización en las redes actuales, empleando tecnologías como OpenFlow y NFV, mejora favorablemente la escalabilidad y el manejo dinámico del tráfico, facilitando que un estudiante pueda activar un firewall virtual en cuestión de segundos usando código.

2.4 REDES DEFINIDAS POR SOFTWARE (SDN)

El surgimiento de las redes definidas por software (SDN) o también conocida en inglés como Software Defined Networking, representa el cambio más relevante en la evolución actual en las telecomunicaciones. Este enfoque busca atender la necesidad de dejar atrás la inflexibilidad de las arquitecturas tradicionales, optando por infraestructuras que son programables, dinámicas y centralizadas.

2.4.1 Definición de las redes SDN: La separación de planos

Según lo estipulado por la ONF (Open Networking Foundation, 2026), el organismo rector de este estándar, SDN es descrita como una estructura en la que el control de la red puede ser programado directamente y se halla físicamente desacoplado de las funciones que se encargan del reenvío. Según (Kurose & Ross, 2021) la esencia de SDN se fundamenta en la división de dos planos que solían trabajar de manera unificada en el hardware:

A. ***Plano de Control (Software):*** Funciona como el cerebro centralizado que determina la dirección de los flujos de datos y la lógica de la red.
B. ***Plano de Datos o Infraestructura (Hardware):*** Desempeña el papel del músculo, realizando el movimiento físico de los paquetes según las indicaciones proporcionadas por el plano de control.

La separación que se logra permite que la inteligencia de la red se desprenda del hardware propietario, convirtiendo dispositivos que antes operaban de forma autónoma en componentes de reenvío simples y programables, según la información descrita por (Mohammad Nowsin et al., 2024)

2.4.2 Evolución histórica: El proyecto Clean Slate y OpenFlow

Las redes SDN no aparecieron de manera aleatoria, sino como un resultado directo de la crisis de escalabilidad enfrentada por los centros de datos entre 2000 y 2010. A mediados de la década de los 2000, el modelo convencional comenzó a "asfixiarse" debido a tres fenómenos:

- **Virtualización de servidores:** Las máquinas virtuales comenzaron a trasladarse de un servidor a otro en cuestión de segundos, mientras que la red permanecía fija y requería modificaciones manuales conexión por conexión.

- **Crecimiento de la nube (Cloud Computing):** Distintas empresas como Google y Amazon, necesitaban volver a configurar miles de conmutadores al instante, lo que era prácticamente imposible con el modelo "caja por caja" característico de las redes tradicionales.
- **Big Data:** Los flujos de información se volvieron masivos e impredecibles.

El momento de revelación se estableció en la Universidad de Stanford con el inicio del programa Clean Slate, liderado por el investigador Nick McKeown, quien junto a sus otros colegas se preguntaron: "Si tuviéramos la oportunidad de reinventar internet desde cero, ¿Cómo lo haríamos?; entonces, se buscó rediseñar el internet desde una "hoja en blanco" para facilitar la prueba de nuevos protocolos sin impactar el tráfico existente. De esta iniciativa surgió OpenFlow en 2008, el primer protocolo estandarizado que permitió transferir el plano de control hacia un servidor externo, creando las bases comerciales y académicas para lo que hoy se conoce como SDN.

2.4.3 Arquitectura funcional de tres niveles

Con el fin de asegurar la escalabilidad y la interoperabilidad, la arquitectura SDN se organiza en tres niveles claramente diferenciados y jerárquicos:

A. Plano de Aplicación

Este es el nivel más alto donde se encuentran los programas que establecen las políticas de la red, como, por ejemplo: Firewalls (cortafuegos), balanceadores de carga o también los sistemas de monitorización. En esta investigación, las aplicaciones aparecerán representadas por scripts de Python que manejan la lógica detrás de los experimentos.

B. Plano de Control (Network OS)

Actúa como el sistema operativo de la red. En el laboratorio virtual, esta función es asumida por un controlador SDN, como ejemplo el controlador Ryu, que posee una visión general de la topología y convierte las necesidades de las aplicaciones en reglas de flujo específicas.

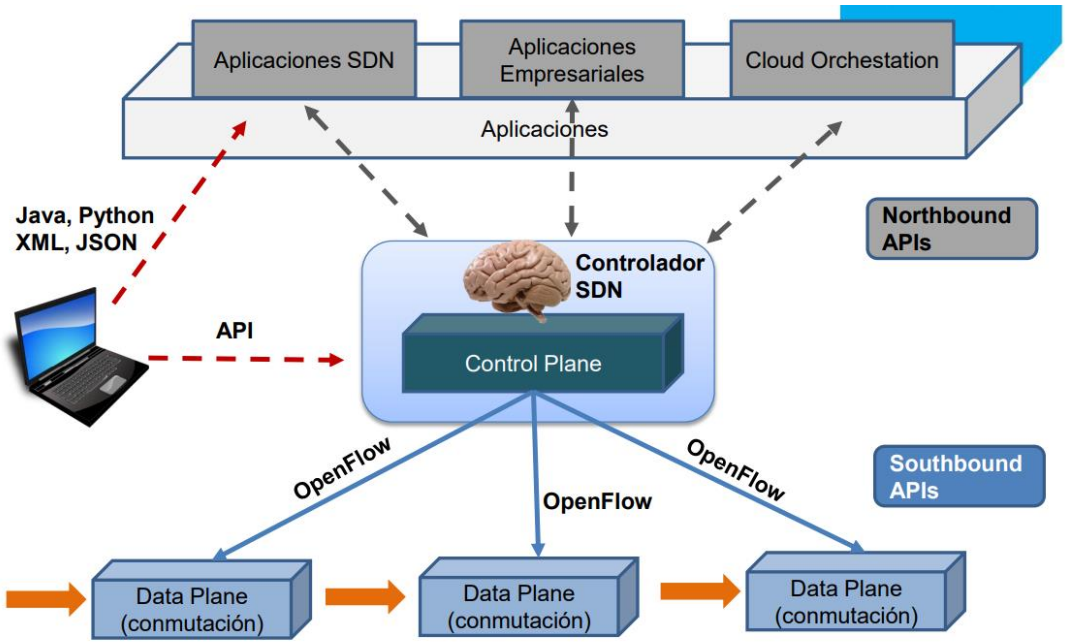
C. Plano de Infraestructura

Este plano está formado por los dispositivos de red, como, por ejemplo; switches físicos o virtuales como Open vSwitch en Mininet. Su trabajo está limitado a

ejecutar instrucciones de reenvío basadas en tablas de flujo, sin tomar decisiones complejas de manera independiente, detalla (Mahmoud Huda, 2026).

Figura 6

Diseño jerárquico de los tres planos de la arquitectura SDN



Nota. Arquitectura jerárquica de SDN que ilustra la separación de los planos de aplicación (superior), control (medio) y datos (inferior), integrados mediante interfaces. Fuente: (Ccoyllo, s.f.)

2.4.4 Interfaces de comunicación: Northbound y Southbound

El funcionamiento unificado de la arquitectura SDN se basa en dos interfaces de comunicación esenciales, que permiten el intercambio de información entre los tres diferentes planos:

A. **Interfaz sur (Southbound Interface - SBI):** Establece la conexión entre el controlador y los conmutadores, este interfaz es el medio por el cual se transmiten las configuraciones y las tablas de flujo. El estándar más utilizado en la industria y el objeto de este estudio, es el protocolo OpenFlow.

B. Interfaz norte (Northbound Interface - NBI): Conecta las aplicaciones con el controlador, normalmente esta interfaz se lleva a cabo a través de API REST, lo que permite que sistemas externos obtengan información de la red o se adicionen políticas de forma dinámica usando formatos comunes como JSON.

Como se puede observar en la Figura 6 existe comunicación entre las interfaces Norte y Sur asegurando la interoperabilidad y la abstracción del hardware.

2.4.5 Análisis comparativo y Optimización

La transición a SDN implica no solo un cambio en la arquitectura, sino también una mejora en las operaciones, la siguiente tabla resume las diferencias clave que respaldan la implementación de este paradigma en contextos modernos y educativos:

Tabla 2

Comparativa técnica entre redes tradicionales y SDN

Característica	Red tradicional	Red definida por software (SDN)
Control	Distribuido (en cada nodo)	Centralizado (en el controlador)
Configuración	Manual / CLI (Caja por caja)	Programable mediante códigos o APIs
Visibilidad	Limitada a dispositivos vecinos	Visión global de toda la topología
Velocidad de cambio	Estática (Semanas / meses)	Dinámica (milisegundos)
Innovación	Atada al fabricante (Lock-in)	Abierta y basada en software.

Nota. La optimización de SDN radica en la abstracción de la lógica, permitiendo que la red responda a tiempo real a las necesidades del tráfico. *Fuente:* Autor.

2.5 PROTOCOLOS DE COMUNICACIÓN SDN

Para que la separación de planos sea exitosa, es necesario tener un mecanismo estandarizado que posibilite el intercambio de comandos, instrucciones y datos entre el controlador (cerebro) y los dispositivos de infraestructura (músculo). Como señalan (Mohammad Nowsin et al., 2024), esta interacción se lleva a cabo a través de interfaces programables, donde la interfaz sur es la responsable de la gestión directa de las tablas de reenvío en los switches.

2.5.1 El protocolo OpenFlow

El protocolo OpenFlow se considera el estándar predominando para la comunicación en la interfaz sur de las redes SDN, originalmente establecido por la ONF (Open Networking Foundation), este protocolo permite que un servidor remoto que toma el papel del controlador, ajuste el funcionamiento de los dispositivos de red, en tiempo real.

En contraste con los dispositivos convencionales que utilizan firmware propietario para determinar el destino de los paquetes, un switch compatible con OpenFlow opera como un dispositivo básico de reenvío. Según (Kurose & Ross, 2021), el switch no toma decisiones por sí solo; simplemente consulta su tabla de flujo para descifrar que acción realizar con cada paquete que recibe. Si no encuentra una regla adecuada, solicita instrucciones al controlador.

Aunque Mininet admite el uso de controladores elementales, es importante comprender la importancia de OpenFlow, ya que es el protocolo base que permite introducir reglas sobre el tráfico, estableces firewalls y crear balanceadores de carga utilizando códigos de Python.

2.5.2 Estructura de las tablas de flujo (Flow Tables)

El funcionamiento de OpenFlow se basa en tablas de flujo, este es su componente esencial y funciona de la siguiente manera; cada switch posee una o más tablas compuestas por entradas de flujo, cada entrada está compuesta por campos de coincidencia (Match Fields) para filtrar el tráfico según una dirección MAC o IP, contadores para estadísticas y acciones para decidir el destino del paquete, de acuerdo a lo indicado por (Toledo Gómez, 2022). Los componentes fundamentales que definen el manejo del tráfico son:

A. Campos de coincidencia (Match Fields)

Son los parámetros que el switch emplea para reconocer un paquete, estos parámetros están relacionados con los encabezados según los protocolos del modelo OSI y TCP/IP:

- **Capa 1:** Es el puerto de ingresos e identifica a través de que interfaz, ya sea física o virtual, el paquete ingresó.
- **Capa 2:** En el punto de enlace de datos, aquí se identifica el origen de las direcciones MAC (dl_src) y destino (dl_dst), también el tipo de ethernet (VLAN ID). Es esencial para la conmutación (switching).
- **Capa 3:** La capa de red, aquí se distinguen las direcciones IP de origen (nw_src) y destino (nw_dst). Esto facilita el enrutamiento.
- **Capa 4:** En la capa de transporte encontramos, TCP o UDP de origen y destino (tp_src, tp_dst). Estos son indispensables para las prácticas de seguridad (Firewall) y la calidad del servicio (QoS).

B. Contadores (Counters)

Son registros estadísticos que se actualizan cada vez que un paquete coincide con una regla. Estos contadores son vitales para el monitoreo de la red, ya que permiten cuantificar el tráfico, la duración de los flujos y también descubrir posibles ataques de denegación en el servicio.

C. Instrucciones o acciones (Actions)

Específica cual debe ser el procedimiento del switch con el paquete que ha cumplido con la regla. Las acciones más frecuentes incluyen:

- **Forward:** Reenviar el paquete a un puerto designado o a todos los puertos.
- **Drop:** Eliminar el paquete de manera silenciosa (base de un firewall).
- **Modify-Field:** Cambiar elementos del encabezado, como la dirección IP o MAC

2.5.3 Mensajería del protocolo OpenFlow

La relación entre el controlador y el conmutador se maneja a través de tipos de mensajes específicos. De acuerdo con (Mohammad Nowsin et al., 2024) estos se dividen en tres grupos según quien inicie la comunicación:

A. Mensajes Controller-to-switch

Estos son generados por el controlador para administrar, gestionar o examinar el estado del conmutador.

- **FLOW-MOD:** Es el mensaje más importante, usado para añadir, cambiar o suprimir reglas en la tabla de flujo.
- **PACKET_OUT:** El controlador instruye al conmutador para que envíe un paquete a un puerto en particular.

B. Mensajes asíncronos

Estos son generados por el conmutador para informar al controlador sobre eventos de la red.

- **PACKET_IN:** Se produce cuando un paquete llega al conmutador y no existe una regla que lo abarque (Table Miss); el conmutador envuelve el paquete y lo remite al controlador, preguntando: "¿Qué debo hacer con esto?". Este mensaje es muy importante en el proceso de identificación de topología (ARP DISCOVERY).

C. Mensajes simétricos

Estos pueden ser generados por cualquiera de las partes, como los mensajes HELLO para establecer la conexión o ECHO para comprobar la latencia y disponibilidad.

2.6 ENTORNOS DE EMULACIÓN Y DESARROLLO

Para llevar a cabo la creación del laboratorio virtual propuesto, se ha optado por un conjunto de herramientas de software de código abierto, que permiten replicar con precisión el comportamiento de una red SDN real. La elección de estas herramientas se basa en criterios de bajo costo, escalabilidad y compatibilidad con estándares de la industria como OpenFlow y Python.

2.6.1 Distinción fundamental: Emulación y Simulación

Para comprobar la arquitectura presentada en esta tesis, es crucial que se establezca la diferenciación técnica entre dos términos que recurrentemente son utilizado erróneamente como sinónimos: simulación y emulación. Según (Torres Quijije & Zuñiga Paredes, 2021), los simuladores y emuladores son instrumentos clave

para recrear redes virtuales, permitiendo analizar el comportamiento del tráfico sin requerir una inversión en infraestructura física costosa.

A. Simulación (El modelo abstracto o teórico)

Un simulador, como en el caso de Cisco Packet Tracer, es un software que reproduce el comportamiento de una red basado en modelos matemáticos y algoritmos establecidos. De acuerdo con la información de (Zúñiga Sánchez et al., 2025) los simuladores no ejecutan el sistema operativo real de los aparatos, sino que solo lo imitan y se obtienen las respuestas mediante modelos programados.

Esta limitación es significativa para la presente investigación, ya que, al enviar un comando ping en un simulador, el software simplemente presenta una respuesta prefinida, sin crear un paquete ICMP real que se desplace a través de una pila de protocolos como en TCP/IP. Esto impide la utilización de analizadores de tráfico profesionales y la interacción con protocolos de control dinámicos.

B. Emulación (La ejecución real)

En contraste con el simulador, un emulador si replica las funciones de un sistema físico al ejecutar el software real en un entorno virtual. Como menciona (Menéndez Regalado & Vera Rendón, 2023), Mininet permite la creación de topologías virtuales con controladores, hosts y switches utilizando el núcleo de Linux, lo que hace posible la emulación de redes SDN reales, con un nivel de fidelidad que los simuladores no logran alcanzar. La emulación posibilita que el sistema operativo se crea que está funcionando sobre el hardware real, permitiendo así, la ejecución de binarios y aplicaciones de red nativas.

2.6.2 Plataforma de emulación Mininet

Mininet se ha establecido como la plataforma de referencia para la investigación y educación en redes SDN, según (Amaya Fariño, 2022) esta herramienta es reconocida por su habilidad para emular redes y evaluar su desempeño en diversos contextos de SDN, destacando por su facilidad de uso y su naturaleza de código abierto.

Arquitectura interna y virtualización eficiente

A diferencia de los hipervisores convencionales que demandan amplios recursos de hardware, Mininet se apoya en Network Namespaces, según lo detallado por (Silva, 2021) esta tecnología de red definida por software, permite crear

infraestructuras virtuales efectivas, donde cada host o switch emulado opera como un proceso independiente dentro del núcleo de Linux con su propia pila de red.

Esto hace posible que en un solo equipo se puedan desplegar centenares de nodos que actúan como dispositivos autónomos. Además, el diseño y simulación de redes en estos contextos permite validar los protocolos antes de su implementación en redes de producción, disminuyendo así, el margen de error.

2.6.3 El controlador SDN: Ryu

En la estructura de SDN, el controlador funciona como el centro de comando de la red. Para este proyecto, se ha elegido Ryu, de acuerdo con (Blanchet et al., 2021), los controladores SDN concentran la inteligencia de la red y posibilitan su administración a través de aplicaciones programables, lo cual es fundamental para la automatización de la red.

Para el manejo del plano de control, se empleará Ryu, un framework de desarrollo basado en componentes de software, ya que se distingue de otros controladores como ONOS o Opendaylight por su arquitectura liviana y su implementación nativa en Python.

Esta elección es estratégica para la enseñanza en ingeniería, dado que Python es el lenguaje predominante en la programabilidad de redes. La estructura de Ryu facilita el desarrollo y la escritura de aplicaciones de red importando librerías específicas:

- **ryu.base:** Núcelo del framework.
- **ryu.controller:** Manejo de eventos OpenFlow, un ejemplo es EventOFPPacketln.
- **ryu.lib.packet:** Librería poderosa para decodificar y generar paquetes (Ethernet, IPv4, ICMP) directamente desde el código.

A. Programabilidad y API REST

Ryu se destaca por su implementación en Python, lo que simplifica la creación de lógica de red a medida. (Mediero Martí, 2021) indica que el análisis de los controladores SDN destaca que la integración de una API REST (Representational State Transfer o Transferencia de Estado Representaciones) permite una comunicación eficiente entre el plano de control y aplicaciones externas, facilitando la supervisión y

ajuste de flujos de tráfico en tiempo real a través del intercambio de datos en formato JSON.

B. Importancia del formato JSON - JavaScript Object Notation

Los datos de la red (topología, estadísticas de puertos, tablas de flujo) se presentan en formato JSON (Notación de Objetos JavaScript); este es un estándar ligero de intercambio de datos que permite, por ejemplo, que la interfaz gráfica de Ryu represente el mapa de la red de un navegador web en tiempo real.

C. Gestión externa

La utilización de APIs REST exhibe la programabilidad de las SDN, permitiendo que el laboratorio sea administrado mediante scripts externos de Python o aplicaciones web de terceros.

2.6.4 Métricas de desempeño

La evaluación de la red emulada no estaría completa sin la medición de ciertos parámetros técnicos, según (Casilimas Fajardo & Cáceres Guevara, 2022), la arquitectura y operación de las redes SDN deben asegurar niveles ideales de eficiencia. Así, en este contexto, se consideran las siguientes métricas:

- **Latencia:** Es el tiempo de retraso de extremo a extremo.
- **Throughput:** La tasa efectiva de transferencia de información.
- **Jitter:** La fluctuación en el tiempo de llegada de los paquetes.

Para recopilar estos datos, se emplean herramientas auxiliares como: Iperf3 para simular tráfico y medir el máximo de ancho de banda posible en redes IP; y también la herramienta Wireshark, para la recolección de paquetes, garantizando que el funcionamiento de la red SDN se alinee con los objetivos de esta investigación.

CAPITULO 3

DESARROLLO E IMPLEMENTACIÓN DE LA SOLUCIÓN

El presente capítulo detalla el procedimiento técnico y metodológico que se empleará en el desarrollo, ejecución y verificación de un laboratorio virtual que utiliza redes definidas por software (SDN). El proceso se adhiere al esquema metodólogo establecido anteriormente y se divide en tres etapas consecutivas: Análisis de requerimientos, diseño de la arquitectura de red e implementación de la solución en una estación de trabajo piloto.

Antes de proceder con la instalación del software, es fundamental determinar si los recursos físicos de la UCSG y de la estación piloto, son adecuados para manejar la carga de trabajo de un entorno emulado, asegurando así, que no existan latencias que pudieran afectar los resultados. El objetivo principal es establecer una plataforma de emulación operativa que permita llevar a cabo prácticas de conectividad y gestión, superando así la limitación del acceso a equipos físicos específicos.

3.1 FASE 1: ANÁLISIS Y DIAGNÓSTICO DE RECURSOS

La fase inicial del proyecto se enfoca en determinar la viabilidad técnica de la solución mediante la comparación entre los requerimientos mínimos de Mininet y los recursos de hardware accesibles en el laboratorio de telecomunicaciones. Con este propósito, se comparan las necesidades del software de emulación Mininet y la capacidad del hardware presente en los equipos, este análisis preliminar garantiza que la infraestructura disponible, pueda manejar la carga de trabajo de la emulación, facilitando así una adecuada replicación de las prácticas de red.

3.1.1 Caracterización del hardware

La caracterización del hardware es esencial para garantizar la viabilidad, sostenibilidad y capacidad de replicar el entorno virtual, este proceso se abordará desde una doble perspectiva: el uso de una estación de trabajo piloto para desarrollar y diseñar la propuesta, y la recopilación de información técnica en el laboratorio de telecomunicaciones de la UCSG, con el objetivo de establecer una base de recursos que sean compatibles.

A. Estación de trabajo piloto

Para la fase de implementación, la estación piloto funciona como el sistema anfitrión (host). Su caracterización es clave, ya que la emulación de redes a través de

Mininet y la gestión mediante el hipervisor VirtualBox requieren capacidades específicas de procesamiento y memoria. Esto asegura que las métricas de la red, como la latencia y el ancho de banda, reflejen con precisión el comportamiento del sistema y no estén influenciadas por limitaciones del hardware.

Un aspecto importante en este análisis es la verificación de las capacidades de virtualización del hardware (Intel VTx o AMD-V), esta característica se encuentra presente en arquitecturas modernas y es fundamental para que el hipervisor pueda acceder directamente a los recursos del procesador, mejorando así la ejecución del sistema operativo invitado y evitando la pérdida de rendimiento del sistema anfitrión.

A continuación, se presenta un análisis comparativo entre las especificaciones técnicas de la estación de trabajo piloto y los requerimientos mínimos del sistema necesarios para el correcto funcionamiento del hipervisor y la herramienta de emulación Mininet:

Tabla 3

Especificaciones técnicas de la estación de trabajo piloto y requisitos del sistema

Componente	**Especificación mínima requerida (Entorno SDN)**	**Especificación de la estación de trabajo piloto**	**Estado**
Procesador CPU	Intel Core i5 de 2 núcleos.	Intel(R) Core (TM) i5-5200U CPU @2.20GHz.	Cumple.
Núcleos / Hilos	2 núcleos / 4 Hilos.	2 núcleos / 4 Hilos.	Cumple.
Memoria RAM	4 GB.	8 GB.	Cumple.
Almacenamiento	20 – 25 GB de espacio libre.	447 GB con 375 GB disponible.	Excede.
Virtualización	Habilitada en BIOS (Intel VT-x).	Habilitada (Confirmado vía Administrador de tareas).	Validado.

Componente	Especificación mínima requerida (Entorno SDN)	Especificación de la estación de trabajo piloto	Estado
Sistema Operativo	Windows 10/11 64 bits (Equipo anfitrión).	Windows 10 HOME 64 bits.	Cumple.
Hipervisor	VirtualBox 6.1 o superior.	Oracle VirtualBox 7.2.2	Instalado.

Nota. Datos obtenidos de la herramienta de diagnóstico del sistema y rendimiento de Windows. *Fuente:* Autor.

Las características mencionadas en la tabla son apropiadas para establecer un entorno SDN utilizando máquinas virtuales, ya que ofrecen los recursos indispensables para asegurar el adecuado y efectivo funcionamiento del sistema. A pesar de que podrían parecer limitadas en relación con configuraciones de alto rendimiento, estas especificaciones son el soporte necesario para que el sistema operativo, el emulador de red y el controlador SDN se ejecuten correctamente en la máquina virtual.

Por lo tanto, se puede concluir que la máquina virtual puede operar de manera apropiada en la estación de trabajo piloto, ya que dispone de recursos suficientes para; manejar la carga de trabajo, de esta manera facilitando el desarrollo, la prueba y la visualización de topologías de red definidas por software sin poner en riesgo la funcionalidad del entorno.

B. Equipo base disponible en el laboratorio de Telecomunicaciones de la UCSG

Para asegurar que la solución sugerida sea aplicable en los laboratorios de la facultad, se llevó a cabo una recolección de datos y una evaluación técnica del hardware. Esta etapa tuvo como propósito verificar que los dispositivos cuentan con el poder de procesamiento suficiente para manejar la carga simultánea del sistema operativo principal, el hipervisor de tipo dos y la red emulada, sin afectar el desempeño.

Durante la evaluación técnica realizada en el laboratorio de telecomunicaciones de la Facultad de Educación Técnica para el Desarrollo (FETD),

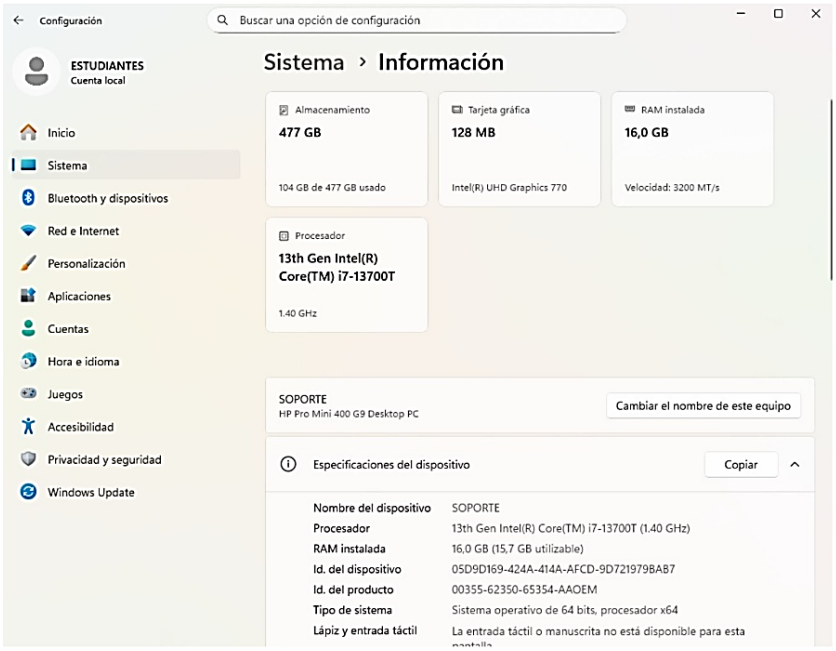
se notó una clara uniformidad en los equipos disponibles. A pesar de la falta de registros administrativos que validen formalmente los estándares tecnológicos de la universidad, la observación en el lugar permitió determinar que las estaciones de trabajo poseen una arquitectura de hardware comparable, con base a esta observación de uniformidad técnica, se realizó la documentación de los recursos de una unidad representativa, asumiendo que sus capacidades operativas son transferibles a otras estaciones en el área.

Este análisis se centró en variables clave, como la capacidad de procesamiento múltiple y la presencia de opciones de virtualización, aspectos importantes que influyen en la estabilidad de la red en la herramienta de emulación Mininet. El estudio mostró que los niveles de CPU, memoria RAM y almacenamiento, son adecuados para permitir la operación simultánea de un controlador SDN y la emulación de conmutadores OpenFlow.

Como sustento técnico del análisis, a continuación, se presentan las evidencias recopiladas directamente de la estación de trabajo analizada, aquí se explican las características que respaldan la posibilidad de la implementación:

Figura 7

Especificaciones generales del sistema anfitrión



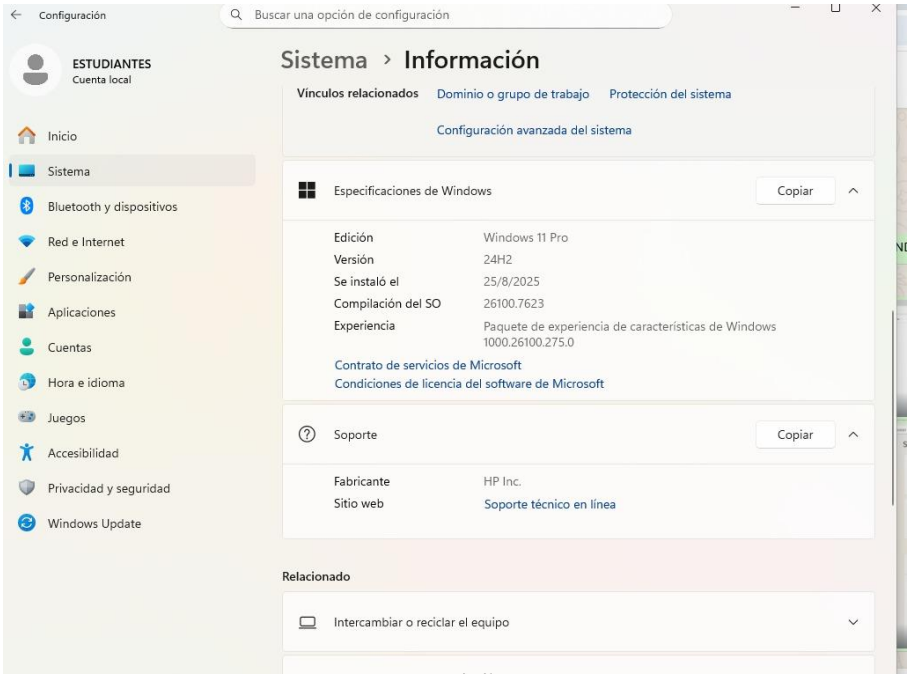
Nota. Captura de pantalla de la configuración del Windows 11 Pro, evidenciando la arquitectura de 64 bits y la memoria física disponible. *Fuente:* Autor

Como se observa en la Figura 7 el ordenador elegido para el desarrollo es un PC de dimensiones compactas (HP Pro Mini 400 G9). Este dispositivo incluye un procesador Intel Core i7-13700T de la décima tercera generación y 16GB de memoria RAM, características que exceden los requerimientos mínimos sugeridos para la virtualización de redes SDN. Esta asignación de recursos permite una segmentación eficiente, garantizando que el hipervisor cuente con suficiente memoria dedicada para el sistema secundario.

Además, es crucial registrar la plataforma de software base. Como se puede ver en la Figura 8, se utiliza el sistema operativo Windows 11 Pro. Esta versión es imprescindible para asegurar una gestión avanzada de controladores y una compatibilidad completa con las extensiones de virtualización más recientes.

Figura 8

Especificaciones del software y compilación del sistema operativo



Nota. El uso de la versión 24H2 asegura que el sistema cuenta con los últimos parches de seguridad y núcleos de ejecución optimizados. *Fuente:* Autor

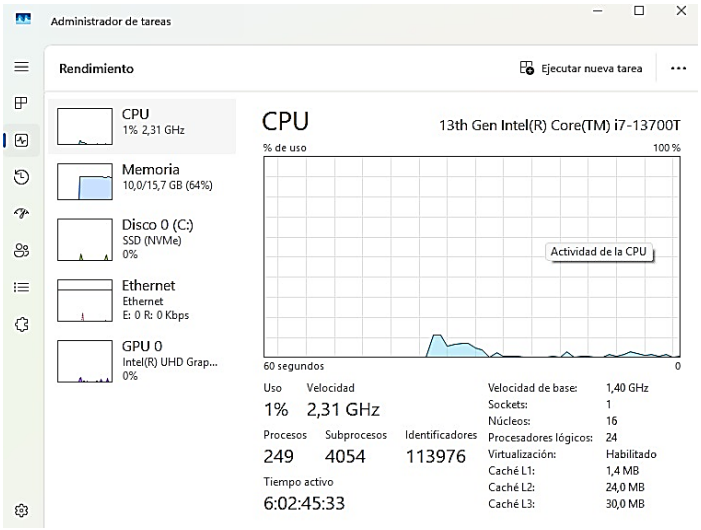
El rendimiento de la emulación con Mininet depende en gran medida de la capacidad del multiprocesamiento, ya que la herramienta genera procesos separados para cada host y conmutador virtual dentro de la red. En este contexto, como muestra la captura de pantalla del administrador de tareas, esta proporciona la información

técnica más relevante: el procesador cuenta con 16 núcleos físicos y 24 hilos de procesamiento.

Esta arquitectura de múltiples hilos permite que Mininet asigne casi todos los recursos a cada nodo de la red virtual. Como se muestra en la ***Figura 9***, la tecnología de virtualización está activada, este aspecto es el requisito técnico más importante, ya que es el que permite que el software VirtualBox tenga acceso directo a las instrucciones del hardware (Intel VT-x) para ejecutar el núcleo de Linux con una mínima latencia.

Figura 9

Monitoreo de rendimiento y estado de virtualización por hardware

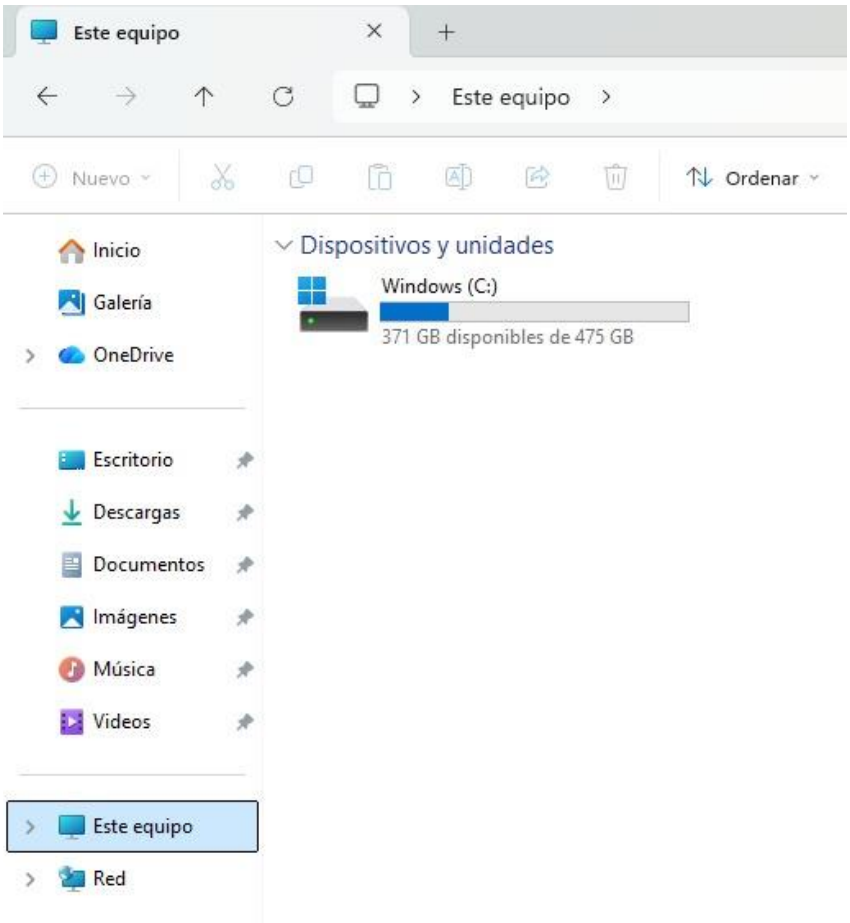


Nota. Captura de pantalla del sistema anfitrión, donde se visualiza que la activación de la virtualización en el firmware (BIOS) es detectada correctamente por el sistema operativo. *Fuente:* Autor.

Finalmente, la capacidad de almacenamiento fue comprobada a través del explorador de archivos, como se observa en la figura 10 el dispositivo tiene un disco de estado sólido (SSD), con tecnología NVMe, que dispone de 371 GB de espacio libre. Esta cantidad es superior a la necesaria para almacenar el sistema operativo invitado Ubuntu, las dependencias de software y los registros de tráfico generados durante las etapas de prueba y validación.

Figura 10

Disponibilidad de almacenamiento en la unidad de estado sólido (C:)



Nota. La baja latencia de los discos SSD NVMe es muy importante para evitar cuellos de botella durante el arranque de la máquina virtual (VM) *Fuente:* Autor.

Para resumir el diagnóstico realizado, se presenta la tabla que consolida el estado de cumplimiento de la estación de trabajo analizada del laboratorio con respecto a las demandas del proyecto:

Tabla 4

Resumen de validación técnica de la estación de trabajo en el laboratorio

Componente	Especificación técnica detectada	Estado de validación
Procesador	Intel Core i7-13700T (1.40 GHz)	Óptimo: 24 hilos lógicos
Memoria RAM	16.0 GB	Óptimo para asignación de 8 GB0020 a VM.
Sistema operativo	Windows 11 Pro (24H2)	Compatible: Edición profesional
Virtualización	Habilitada en BIOS	Cumple: Soporte VT-x

Nota. La validación de "óptimo" indica que el componente no solo cumple con el requisito mínimo, sino que también posee la capacidad adicional necesaria para tareas de monitoreo y análisis de datos en tiempo real. *Fuente:* Autor.

C. Análisis de suficiencia y brecha técnica.

Una vez culminado el levantamiento de información, se revisó el contraste entre los requerimientos de software y el hardware diagnosticado, con el fin de elaborar una comparación técnica que asegura la viabilidad del entorno virtual. A continuación, se detallan las conclusiones de los datos recopilados:

Capacidad de procesamiento

Aunque el emulador Mininet demanda al menos dos núcleos para realizar sus funciones básicas, la estación de trabajo analizada del laboratorio, ofrece una arquitectura con 24 hilos lógicos. Esta capacidad para manejar múltiples procesos es significativa, ya que la herramienta asigna tareas distintas a cada anfitrión y conmutador virtual. Esta disponibilidad de recursos garantiza una gran fidelidad durante las pruebas de estrés de red.

En cuanto a la estación de trabajo piloto, que cuenta con un procesador Intel Core i5, es adecuada para llevar a cabo las guías de práctica previstas, aunque es aconsejable no sobrepasar la emulación de diez conmutadores a la vez para evitar latencias provocadas por el CPU.

Gestión y optimización de la memoria RAM

El entorno virtual propuesto tiene un consumo aproximado de 2.5 GB en reposo, la estación de trabajo analizada cuenta con 16 GB de RAM, puede destinar 8GB exclusivamente a la máquina virtual, dejando así una suficiente capacidad para que el sistema base ejecute simultáneamente el controlador Ryu y herramientas de análisis de tráfico como Wireshark.

Para la estación de trabajo piloto que cuenta con 8 GB, la solución es técnicamente viable, si se limita la ejecución de aplicaciones no esenciales en el sistema operativo base, para de esta manera poder evitar el uso de memoria de intercambio, lo que afectaría la precisión de las métricas de latencia.

Rendimiento de Entrada/Salida y almacenamiento

Un aspecto bastante importante que se ha identificado es la adopción de unidades de estado sólido; la estación de trabajo del laboratorio utiliza tecnología NVMe, lo que garantiza que el intercambio de datos entre la memoria temporal y el disco virtual no cause cuellos de botella.

Esta tasa alta de transferencia permite que tanto el inicio del sistema invitado, como la ejecución de scripts de topología en Python, se realicen en cuestión de segundos. En el laboratorio se ha comprobado que el espacio es adecuado, aunque la velocidad de respuesta puede fluctuar según el tipo de unidad (HDD o SSD) en cada estación.

Validación de virtualización y portabilidad

Se ha verificado que ambos entornos tienen habilitado el soporte para la virtualización por hardware (VT-x), este aspecto permite que el hipervisor ejecute instrucciones de anillo 0, eliminando la carga de la emulación por software. Esta compatibilidad técnica garantiza la portabilidad de la solución; la coincidencia en la disponibilidad de capacidades de virtualización en ambos, el equipo piloto y la infraestructura existente en el laboratorio de telecomunicaciones de la UCSG, asegura

que la solución se pueda replicar completamente. Esto permite que los estudiantes de telecomunicaciones realicen prácticas de red de manera eficiente, cumpliendo con los estándares de rendimiento requeridos para la validación de arquitecturas SDN.

D. Entorno físico y hardware de soporte

La implementación del laboratorio virtual está programada para ser realizada en las instalaciones de la Facultad de Educación Técnica para el Desarrollo (FETD). Es importante entender el espacio físico donde se llevará a cabo el cambio de prácticas que utilizan hardware tradicional a un entorno emulado.

En la Figura 11 se muestra una vista general del laboratorio de telecomunicaciones de la UCSG, este lugar actúa como el entorno controlado donde se verificará la solución, equipado con la infraestructura eléctrica y de red necesaria. Aun así, para los objetivos de este trabajo, el enfoque se dirige a la capacidad de procesamiento interno de estas terminales.

Figura 11

Entorno físico del laboratorio de telecomunicaciones - FETD

Nota. Vista general del laboratorio donde se aprecia la disposición de las estaciones de trabajo y la infraestructura de soporte para las prácticas de redes LAN. *Fuente:* Autor

3.1.2 Análisis de requerimientos del software Mininet

Para cumplir con el primer objetivo específico se ha realizado un análisis detallado de las necesidades funcionales y técnicas que debe cumplir la herramienta de emulación. En el campo de telecomunicaciones es importante decidir entre un simulador y un emulador: un simulador representa el comportamiento a través de algoritmos, mientras que Mininet opera como un emulador que utiliza la virtualización del sistema operativo para ejecutar código real del núcleo de Linux, asegurando una precisión técnica equivalente al hardware físico.

A continuación, se clasifican los requerimientos para garantizar la operatividad en la estación de trabajo piloto y en el laboratorio de telecomunicaciones:

A. Requerimientos funcionales (RF)

- **RF1 - Emulación de dispositivos de capa 2 y 3:** El sistema debe ser capaz de crear nodos que simulen hosts y switches, cada uno con su propia pila TCP/IP, tablas de enrutamiento y direcciones MAC exclusivas.
- **RF2 - Soporte del protocolo OpenFlow:** El emulador debe incluir de manera nativa la compatibilidad con OpenFlow (v1.0 y v1.3), facilitando la interacción con el controlador SDN.
- **RF3- Gestión de topologías dinámicas:** La herramienta debe permitir crear arquitecturas de estrella y árbol, usando scripts de Python, haciendo posible ajustar parámetros como el ancho de banda y el retardo.
- **RF4 - Interacción con el Kernel:** Es necesario que cada host virtual tenga la capacidad de ejecutar aplicaciones de red reales como Wireshark, ping e iperf.

B. Requerimientos no funcionales (RNF)

- **RNF1 - Bajo consumo de recursos:** La solución debe funcionar eficientemente en la estación piloto y las estaciones de trabajo en el laboratorio de telecomunicaciones, sin afectar el rendimiento del sistema anfitrión.
- **RNF2- Portabilidad:** La solución debe ser empaquetada en un formato ".OVA" para su distribución en la FETD.

- **RNF3 - Licenciamiento:** Se prioriza el uso de software de código abierto, para así evitar costos de adquisición para la universidad.

3.1.3 Arquitectura del motor Mininet – Fundamentos técnicos

La selección de Mininet se basa en su arquitectura que utiliza Network Namespaces (Espacios de nombre de red); a diferencia de las máquinas virtuales pesadas, Mininet emplea virtualización ligera del núcleo de Linux. En la implementación para el laboratorio, esto permite que cada host virtual funcione como un sistema operativo autónomo, con su propia tabla de rutas e interfaces.

Al ejecutarse en el hardware HP Pro Mini 400 G9 mencionado anteriormente, el emulador utiliza la aceleración de hardware para llevar a cabo las siguientes funciones esenciales:

A. Aislamiento de recursos

Cada nodo de la red emulada opera en un entorno de memoria aislado, evitando que haya conflictos de procesos entre los diferentes hosts virtuales.

B. Conmutación virtual mediante Open vSwitch (OVS)

Mininet incorpora OVS para funcionar como el conmutador de datos. Este software puede manejar tramas Ethernet a velocidades cercanas a la línea física, permitiendo que las prácticas de redes LAN se mantengan con una precisión del 99% en comparación con un conmutador físico.

Se ha elegido este elemento porque es el referente en el ámbito de redes virtuales, lo que posibilita que el alumno ejercite la segmentación de redes LAN y la gestión de flujos de datos a través de la programación. Esta configuración favorece la división entre el plano de control (donde se encuentra la lógica) y el plano de datos (por donde fluye el tráfico), que es un principio esencial en todo el contexto de las redes definidas por software (SDN)

C. Conmutación con controladores SDN

Ya que la estructura de Mininet permite diferenciar el plano de control del plano de datos, en este contexto, esto resulta muy importante ya que posibilita la conexión de la red emulada a un controlador externo, en este caso se usa el controlador Ryu, utilizando el protocolo OpenFlow.

3.1.4 Selección y justificación de la pila tecnológica

Luego de evaluar la compatibilidad en la estación de trabajo piloto y en el laboratorio de telecomunicaciones, se eligieron las siguientes versiones de software, respaldadas por su estabilidad y soporte a largo plazo (LTS):

A. Hipervisor tipo II: VirtualBox (7.2.2)

Este hipervisor fue elegido por su solidez al manejar adaptadores de red del tipo "puente" y su capacidad para activar instrucciones Intel VT-x, minimizando la latencia en la simulación de paquetes; se prefirió VirtualBox sobre otras opciones como VMware, Workstation, KVM, por su:

- **Flexibilidad de red:** Permite establecer múltiples adaptadores virtuales, esencial para diferenciar el tráfico de gestión (NAT) del tráfico de datos del laboratorio (Host-Only).
- **Soporte para virtualización de hardware:** Su facultad de interactuar con las instrucciones Intel VT-x permite que la máquina virtual de Ubuntu pueda procesar redes con una latencia reducida.

B. Sistema operativo (Ubuntu 22.04.5 LTS Desktop)

Se eligió la versión de escritorio con interfaz gráfica para simplificar el uso de herramientas visuales como Wireshark. La opción LTS asegura actualizaciones de seguridad y estabilidad del núcleo hasta el 2027, garantizando que el laboratorio permanezca actualizado, la decisión de usar esta distribución de Linux se basa en las siguientes consideraciones técnicas:

- **Estabilidad del núcleo:** Mininet depende considerablemente del núcleo de Linux para el intercambio de paquetes. Ubuntu 22.04 presenta un núcleo de soporte a largo plazo (LTS) que ha sido extensivamente validado en contextos de red.
- **Ecosistema en repositorios:** La facilidad para instalar dependencias esenciales como Open vSwitch, Python 3-pip y bibliotecas de desarrollo de redes a través del gestor apt disminuye la complejidad del manual de instalación para el alumno.

C. Mecanismo de conmutación Open vSwitch (OVS)

Dentro de Mininet, el componente responsable de la transferencia de datos es Open vSwitch, su incorporación es necesaria por:

- **Norma industrial:** Es el conmutador virtual de código abierto más empleado en centros de datos y ambientes de nube.
- **Soporte mejorado para VLANs:** Facilita que los estudiantes trabajen con la segmentación de redes LAN mediante la etiquetación de tramas 802.1Q, un concepto fundamental en la carrera de telecomunicaciones.

D. Emulador (Mininet 2.3.0)

Esta versión destaca por su estabilidad, ofreciendo soporte nativo para Python 3, permitiendo que los scripts desarrollados sean modernos y fáciles de mantener, esto transforma la creación de topologías en un proceso de desarrollo de software moderno, ya que, al ser scripts escalables, la API de Python permite definir distintas topologías personalizadas (árbol, estrella) mediante objetos y clases, facilitando la creación de escenarios de red complejos con pocas líneas de códigos.

Mininet 2.3.0 trabaja estrechamente con Open vSwitch y actúa como el motor de reenvío de datos. En el contexto de la emulación de enlaces, esta versión mantiene una configuración precisa de parámetros de red, tales como el ancho de banda, el retardo y la pérdida de paquetes a través de diversas herramientas internas de Linux. Además, optimiza la conexión hacia controladores remotos; ya sea que se utilice en controladores locales o uno alojado en una máquina virtual distinta. También, garantiza una latencia mínima en el plano de control, lo que resulta bastante importante para poder obtener resultados precisos en esta investigación, donde el tiempo de respuesta es también una variable de estudio.

E. Controlador SDN (RYU framework)

Se optó por RYU en lugar de ONOS o OpenDaylight por su diseño en Python, lo cual facilita el aprendizaje para los estudiantes y proporciona una visualización más clara de los flujos de la red. A continuación, se presenta una breve comparativa que justifica su elección frente a otros controladores populares:

Tabla 5

Comparativa de controladores SDN

Criterio	RYU	ONOS / OpenDaylight	Justificación técnica
Lenguaje de programación	Python	Java	Python facilita la curva de aprendizaje y la redacción de guías de práctica de manera rápida
Consumo de memoria	Bajo	Alto	ONOS requiere al menos 4-8 GB de RAM solo para el controlador, lo que saturaría la estación de trabajo piloto.
Arquitectura	Basada en componentes	Basada en OSGi	RYU es más lineal y permite ver el flujo de los paquetes de forma más transparente.
Propósito de la tesis	Académico / Experimental	Industrial / Producción	RYU se ajusta mejor al diseño de laboratorios de aprendizaje práctico.

Nota. La elección de RYU garantiza que el laboratorio sea ejecutado de forma exitosa, incluso en equipos que cuenten con limitaciones en la memoria. *Fuente:* Autor.

La unión de todos estos componentes conforma la pila tecnológica que se usará en el proyecto, a continuación, se resumen los aspectos más importantes de la línea base de la implementación:

Tabla 6

Consolidación de la pila tecnológica seleccionada

Componente	Versión	Justificación
Sistema operativo invitado	Ubuntu 22.04.5	Estabilidad del kernel y compatibilidad con drivers gráficos.
Controlador	Ryu	Bajo consumo de RAM y facilidad de programación en Python, priorizando la claridad pedagógica sobre una escala industrial.
Switch virtual	Open vSwitch	Soporte avanzado de protocolos de capa 2 y OpenFlow.
Lenguaje	Pyhton 3.x	Estándar actual para la automatización de infraestructura de red.
Emulador	Mininet 2.3.0	Soporte nativo de Python 3 y estándares de virtualización ligera.

Nota. Consolidación de herramientas de software de código abierto, integradas para la creación del entorno SDN002, priorizando versiones con soporte extendido y una compatibilidad nativa con protocolos programables. *Fuente:* Autor.

3.2 FASE 2 – DISEÑO DE LA SOLUCIÓN

Esta fase se relaciona directamente con el cumplimiento del segundo objetivo específico, se inicia con la organización lógica y el diseño arquitectónico completo del laboratorio virtual. Este procedimiento de diseño tiene como meta convertir de forma efectiva los requerimientos técnicos y las necesidades educativas que se identificaron en el diagnóstico previo en modelos operativos lógicos; estos modelos permitirán realizar experimentos prácticos sin necesidad de una infraestructura física que sea costosa o difícil de acceder.

La planificación arquitectónica realizada proporciona una representación precisa de los elementos de red, definiendo no solo la disposición estratégica de los conmutadores virtuales y las conexiones de comunicación, sino también el riguroso esquema de asignación de direcciones IP que resulta esencial para garantizar un flujo adecuado y la segmentación de los datos en el espacio virtual.

Desde una perspectiva de ingeniería, la viabilidad de este diseño se fundamenta en la alta precisión que ofrecen las plataformas actuales de emulación; al llevar a cabo la implementación de las arquitecturas bajo el enfoque de las redes SDN, se asegura que la gestión de recursos se realice de manera centralizada y dinámica, la cual ofrece una flexibilidad operativa que falta en los diseños tradicionales. En estudios recientes, como los de (Alfaify et al., 2023), apoyan el uso de herramientas como Mininet para crear y validar complejas topologías de red, garantizando que los datos de rendimientos y los resultados de conectividad obtenidos sean un reflejo confiable de lo que se podría encontrar en entornos físicos de producción.

Además, según (Askar & Keti, 2021) quienes apoyan el uso de herramientas como Mininet para crear y validar complejas topologías de red, garantizando que los datos de rendimiento y los resultados de conectividad obtenidos sean un reflejo confiable de lo que se podría encontrar en entornos físicos de producción, así como también, la capacidad para un rápido prototipado en Mininet permite que estos diseños complicados sean validados con una alta precisión. De esta manera, el diseño propuesto se establece como una herramienta educativa excepcional que fortalece el aprendizaje práctico dentro de la carrera de telecomunicaciones de la UCSG.

3.2.1 Modelado de Topologías

El modelado de las topologías lógicas es el eje operativo de la propuesta, fundamentándose en la necesidad esencial de replicar ambientes LAN reales que faciliten a los estudiantes la transición fluida desde configuraciones simples hasta arquitectura jerárquicas más complejas. Con una planificación arquitectónica detallada, se puede lograr la abstracción de los componentes de red, estableciendo la disposición estratégica de los conmutadores virtuales y las conexiones de comunicación bajo un esquema de asignación de IP estricto, que garantiza la integridad y la clasificación del tráfico experimental.

La eficacia de este modelado se basa en el uso de la capacidad de Mininet para emular hardware real a través de procesos aislados dentro del núcleo de Linux, lo que asegura una calidad técnica superior a la proporcionada para los simuladores tradicionales, permitiendo la ejecución de código real de red.

Bajo este enfoque, se han delineado dos escenarios experimentales distintos que funcionan como laboratorios prácticos de aprendizaje. Ambos diseños se integran en el paradigma SDN, confiando la inteligencia de la red al controlador Ryu para ofrecer a los estudiantes, una comprensión clara de la diferenciación entre el plano de control y el plano de datos.

A. Diseño de topología 1: Red LAN en estrella

En este primer ejemplo se presenta la base fundamental de las redes de área local actuales, sirviendo como punto de referencia operacional para justificar la implementación del entorno virtual. La estructura se basa en una topología radial, en el que un único conmutador virtual central, llamado s1, que está basado en el estándar Open vSwitch (OVS), desempeña el papel importante del núcleo de datos.

A este dispositivo se conectan cuatro nodos o terminales finales, también llamados hosts - desde h1 hasta h4- a través de conexiones virtuales de baja latencia que utilizan Network Namespaces del núcleo de Linux, asegurando un aislamiento completo entre las diferentes estaciones.

Para este laboratorio se ha definido el segmento de red privada 10.0.1.0/24 lo que permite a los estudiantes, realizar su trabajo en un entorno controlado que aísla el tráfico experimental, evitando interferencias con otros procesos en el sistema anfitrión.

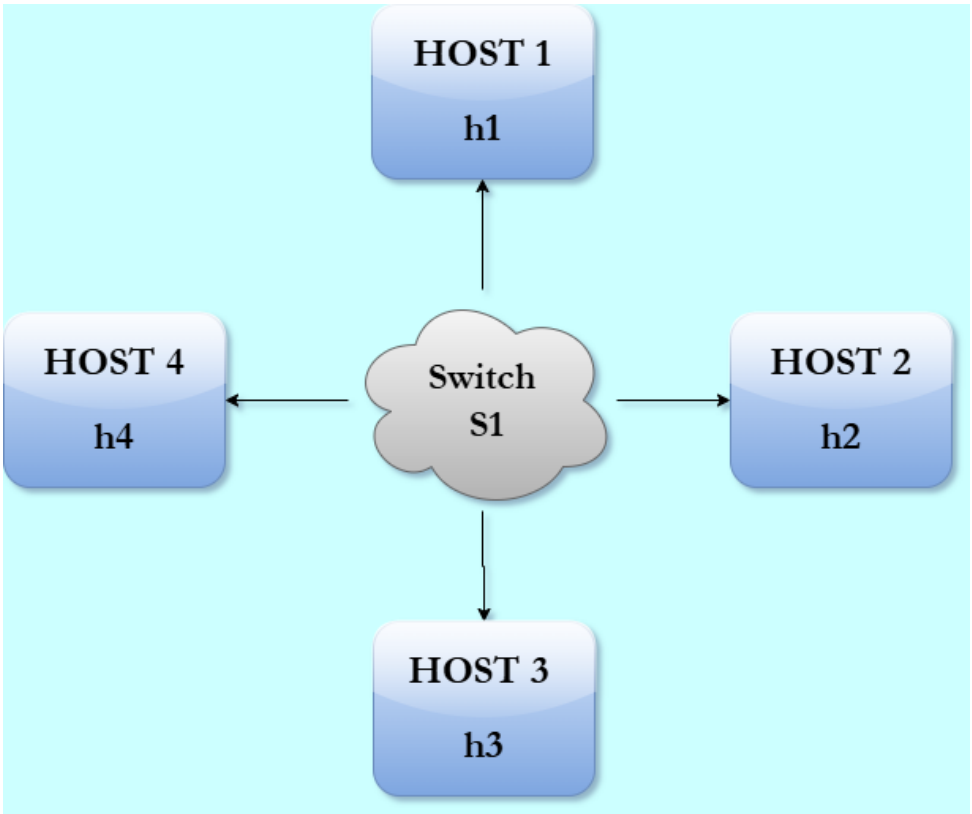
Siguiendo estrictamente el modelo de SDN, la arquitectura de este entorno lleva a cabo una separación total de los planos; el interruptor central no tiene inteligencia de conmutación local, funcionando únicamente como un dispositivo de reenvío que utiliza tablas de flujo.

Según este sistema de conmutación reactiva, cuando se detecta un paquete con una dirección MAC de destino desconocida, lo que hace el interruptor no es decidir el enrutamiento por sí mismo; en lugar de eso, realiza el encapsulamiento de la cabecera y la envía al controlador remoto Ryu mediante un mensaje Packet-In usando el protocolo OpenFlow 1.3.

Esta interacción permite que la lógica de la red sea programable y centralizada, dándole a los estudiantes la posibilidad de observar en directa cómo el cerebro de la red introduce de manera dinámica las reglas necesarias para gestionar el tráfico.

Figura 12

Diagrama lógico del escenario 1: Red en Estrella - SDN



Nota. El diseño emplea conexiones de comunicación de punto a punto virtuales con parámetros de retraso cero para simular un entorno de laboratorio ideal. *Fuente:* Autor.

A continuación, se resumen los pasos del diseño lógico:

- **Establecimiento del núcleo:** Se configura un único interruptor virtual central de tipo Open vSwitch (s1), que actuará como el centro de convergencia para la información.
- **Integración de estaciones finales:** Cuatro computadoras o host terminales (h1, h2, h3, h4) se conectan directamente al interruptor central mediante enlaces virtuales.
- **Asignación de control:** A diferencia de las redes tradicionales, el interruptor s1 no toma decisiones de manera independiente, la inteligencia de la red se delega a un "cerebro" externo, llamado Controlador Ryu, cumpliendo con el paradigma de las redes SDN.

Desde un punto de vista pedagógico, este escenario ha sido diseñado para permitir que el estudiante pueda entender y examinar el concepto de un dominio de difusión (en inglés conocido como broadcast domain) único desde un enfoque técnico detallado.

Al realizar pruebas de conectividad de extremo a extremo, los estudiantes van a poder emplear herramientas de monitoreo como Wireshark para analizar la estructura de los paquetes y comprobar como interactúan las capas 2 y 3 del modelo OSI, enfocándose en dos protocolos clave:

- **Protocolo ARP (Asignación de direcciones):** Este protocolo registra el procedimiento mientras capta el instante preciso en que se procesan las solicitudes de difusión por el controlador, facilitando la representación de la topología de la red virtual.
- **Protocolo ICMP (Mensajes de Control):** Se utiliza para efectuar pruebas de conectividad (ping), asegurando así la comunicación Unicast.

B. Diseño de topología 2: Red jerárquica en árbol

Este segundo escenario se ha diseñado con el objetivo de explorar conceptos avanzados relacionados con la escalabilidad y la gestión del tráfico, replicando la estructura de red típica en universidades y grandes empresas. A diferencia de la red plana en estrella, esta configuración se organiza de manera jerárquica con una profundidad de dos niveles (Depth= 2) y un factor de ramificación de dos (Fanout= 2),

permitiendo que los estudiantes puedan interactuar con tres capas de funciones claramente definidas en el proceso del diseño lógico:

- **Primer nivel – Capa core o núcleo:** Se establece un switch raíz (s1) en la parte superior del árbol, su responsabilidad es interconectar las diferentes ramas de la red.
- **Segundo nivel – Capa de agregación:** Se instalan dos switches secundarios (s2 y s3) que se conectan directamente al switch raíz. Su rol es agrupar el tráfico de los niveles inferiores.
- **Tercer nivel – Capa de acceso:** Se sitúan cuatro hosts en la base del árbol. Las computadoras h1 y h2 están conectadas al switch 2 (s2), esta vendría a ser la Rama A; mientras que h3 y h4 se conectan al switch 3, que da a la Rama B.

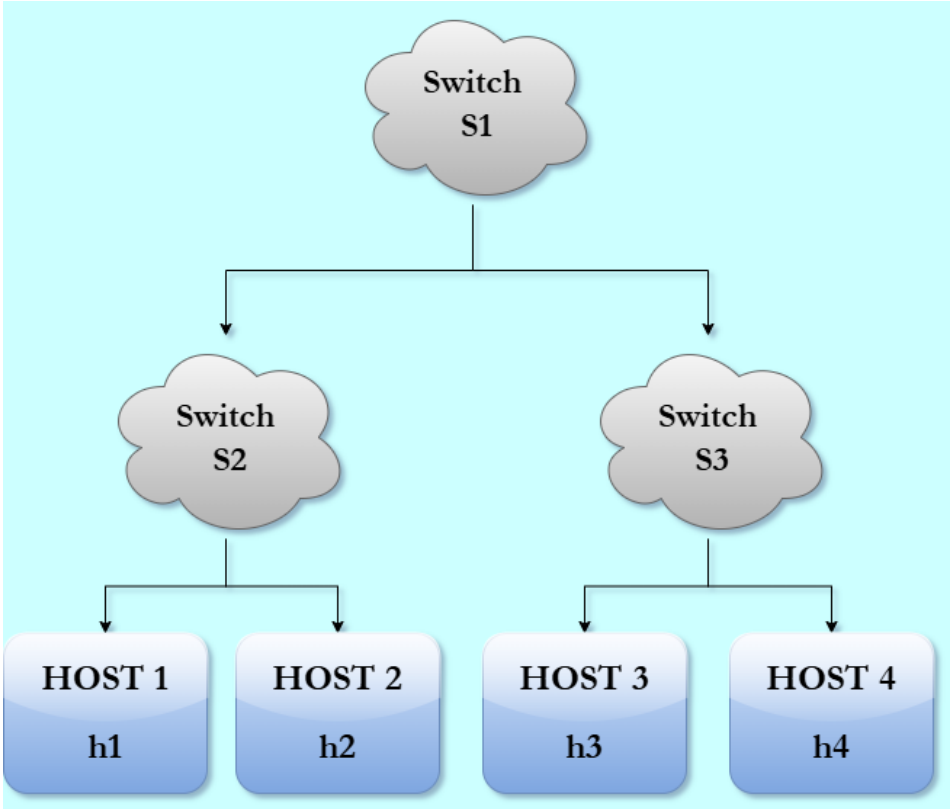
La implementación de esta topología bajo el enfoque SDN demuestra la ventaja del control centralizado frente a las limitaciones de las redes tradicionales. En una red jerárquica convencional, la presencia de múltiples caminos físicos a menudo requiere el uso del protocolo STP (Spanning Tree Protocol) para prevenir tormentas de difusión y bucles lógicos, lo que frecuentemente resulta en enlaces subutilizados. Sin embargo, en este entorno emulado, el controlador Ryu cuenta con una perspectiva global y actual de toda la topología, lo que le permite calcular y crear rutas óptimas de forma dinámica y multi-salto sin el riesgo de bucles, maximizando así, el uso de cada enlace físico disponible.

Desde el enfoque de evaluación del rendimiento, la estructura jerárquica está diseñada para llevar a cabo pruebas de resistencia y análisis de puntos críticos en los enlaces principales; al emplear herramientas especializadas como iperf, los estudiantes pueden saturar de tráfico los vínculos entre las capas de acceso y agregación, con el fin de calcular métricas esenciales, incluyendo el Throughput, que es la tasa de transferencia efectiva y el Jitter, el cual es la variación de la latencia.

Esta actividad práctica es fundamental para comprobar la solidez de la red y el funcionamiento del controlador en condiciones de alta demanda, permitiendo la medición de retardos de ida y vuelta (RTT) que no podrían detectarse en redes de un solo nivel, aumentando así la capacidad para diagnosticar y optimizar infraestructuras actuales.

Figura 13

Diagrama lógico del escenario 2: Diseño de árbol jerárquico



Nota. La jerarquía facilita el estudio de la propagación de paquetes a través de varios saltos de conmutación. *Fuente:* Autor.

3.2.2 Planificación del direccionamiento IP y segmentación lógica

La consolidación de los modelos arquitectónicos que se han propuesto, no es simplemente una disposición de nodos, sino que requiere la ejecución de un esquema de direccionamiento estricto que permita la escalabilidad a largo plazo, que también facilite una identificación eficiente de problemas, y garantice una clara separación del tráfico experimental entre los diferentes escenarios planteados.

Para que las estructuras topológicas de estrella y árbol sean eficientes, es necesario implementar un modelo de direccionamiento que se cumpla con los estándares internacionales del mundo de las telecomunicaciones. El direccionamiento que inicia con 10.x.x.x proviene del RFC 1918 (Address Allocation for Private Internets). Este documento técnico establece que este rango se reserva únicamente para redes privadas, lo que significa que estas direcciones no pueden ser rastreadas en la internet pública. Esto tiene bastante relevancia en el laboratorio virtual, ya que

garantiza que este pueda funcionar dentro de los dispositivos de la facultad sin interferir con la red WI-FI o la red administrativa de la universidad.

A. El default de Mininet

Por defecto, el emulador Mininet, asigna a cada host una dirección en el rango 10.0.0.0/8, sin embargo, en este estudio se ha decidido dividir ese bloque extenso en segmentos más pequeños utilizando la máscara /24 (255.255.255.0):

- **Segmento 10.0.1.0/24 (Topología Estrella):** Se elige para simbolizar una red local típica de oficina, usando el número .1 en el tercer octeto, identificamos visualmente que corresponde al laboratorio 1.
- **Segmento 10.0.2.0/24 (Topología Árbol):** Al utilizar el número .2, se separa lógicamente el tráfico, si un estudiante encuentra una IP que inicia con 10.0.2.x, reconocerá de inmediato que el paquete pertenece a la topología de árbol y no a la de estrella.

El rango 10.0.1.0/24 proviene del conjunto de direcciones privadas establecido en el RFC 1918, se ha elegido el tercer octeto para distinguir entre diferentes configuraciones de prueba: el número "1" representa la topología en estrella y el número "2" indica la topología en árbol. Esta segmentación lógica, que supera la configuración plana determinada de Mininet, facilita una mejor gestión del tráfico y proporciona un entorno de aprendizaje más alineado con las realidades de las redes empresariales.

Con esta base se ha diseñado un esquema completo utilizando el estándar IPv4 con el bloque de direcciones de red privada 10.0.0.0, asegurando que el entorno virtual creado con herramientas como Mininet sea una representación confiable y técnicamente valida de los escenarios reales de telecomunicaciones.

Para llevar a cabo esta planificación de una manera estructurada y coherente, la arquitectura técnica del entorno simulado se divide en cuatro aspectos clave que abarcan desde el aislamiento lógico del tráfico hasta la identificación precisa de los elementos de la red:

B. Justificación de la máscara /24 frente a la /8

Aunque la configuración predeterminada de Mininet utiliza un prefijo de enrutamiento /8 para el entorno virtual, este diseño ha optado por una segmentación

rigurosa empleando una máscara de subred de longitud fija (FLSM) de 255.255.255.0 (/24). Esta parametrización deliberada tiene como objetivo imitar las prácticas comunes en la gestión de redes locales.

La elección técnica se basa en dos metas principales:

Dominio de difusión

Con una máscara /8, se podrían incluir más de 16 millones de dispositivos en una única red, lo que provocaría ineficiencia y grandes cantidades de tráfico no deseado; utilizando una máscara /24, se limita el número de direcciones utilizables a un máximo de 254 por segmento.

Esta limitación es fundamental en entornos virtualizados, ya que evita la sobrecarga de procesos en el núcleo del sistema operativo anfitrión debido a posibles tormentas de difusión, lo que es ideal para que los estudiantes aprendan a determinar rangos de red, puertas de enlace (gateways) y direcciones de difusión (broadcasts) en un entorno controlado, que no sature la CPU del equipo piloto.

Segmentación lógica de escenarios

Se ha asignado el segmento 10.0.1.0/24 exclusivamente para la topología en estrella y el segmento 10.0.2.0/24 está reservado para la topología en árbol, esta separación es esencial para prevenir colisiones de paquetes o conflictos de direcciones IP, asegurando que los estudiantes puedan examinar los protocolos de manera individual.

C. Identificadores para SDN: ID de Datapath (DPID)

La asignación de identificadores únicos permite que la inteligencia de la red sea verdaderamente programable y centralizada.

- **Papel del ID de Datapath en OpenFlow 1.3:** El plano de control usa el DPID para reconocer de manera única cada dispositivo de conmutación, este identificador es muy importante para que la interfaz sur pueda enviar configuraciones y tablas de flujo específicas a cada nodo.
- **Asignación de identificadores por escenario:** Se han establecido identificadores específicos para asegurar una organización metódica,

por ejemplo; en el escenario 1, el conmutador central s1 se designa con el DPID 1. En el escenario 2, el conmutador raíz de la capa central s1 recibe el DPID 10, por otro lado, los dispositivos en la capa de agregación (s2 y s3) operan con los DPID 20 y 30 respectivamente.

- **Importancia para el reconocimiento automatizado:** Al poner en marcha el entorno simulado, el controlador se apoya en los DPID para elaborar un mapa lógico global de la red, facilitando así, la introducción reactiva de flujos y la gestión dinámica del tráfico.

D. Esquemas de asignación y mapeo de puertos

La organización metódica de los puntos de conexión es un pilar muy importante para asegurar la total consistencia entre el diseño lógico y la infraestructura visualizada. A continuación, se detallan formalmente los esquemas de asignación que regulan la ejecución de ambas topologías.

Mapeo del escenario 1: Topología estrella

En la topología estrella se ha establecido una relación directa y secuencial para facilitar el proceso de aprendizaje y la incorporación de flujos reactivos; el conmutador principal (s1) utiliza una interfaz de bucle local para su operación interna, mientras que los cuatro hosts finales se conectan de manera organizada a los puertos virtuales del conmutador.

Tabla 7

Planificación IP y mapeo de puertos - Escenario 1: Topología estrella

Dispositivo	Interfaz física / virtual	Puerto en Switch s1	Dirección IP lógica	Máscara de subred
Switch s1	1o:0	N/A (DPID: 1)	127.0.0.1	255.0.0.0
Host h1	h1-eth0	s1-eth1 – Puerto 1	10.0.1.1	255.255.255.0

Dispositivo	Interfaz física / virtual	Puerto en Switch s1	Dirección IP lógica	Máscara de subred
Host h2	h2-eth0	s1-eth2 – Puerto 2	10.0.1.2	255.255.255.0
Host h3	h3-eth0	s1-eth3 – Puerto 3	10.0.1.3	255.255.255.0
Host h4	h4-eth0	s1-eth4 – Puerto 4	10.0.1.4	255.255.255.0

Nota. La asignación de los puertos virtuales se realiza de forma secuencial, el segmento 10.0.1.0/24 aísla la comunicación en este escenario base. *Fuente:* Autor.

Mapeo del escenario 2: Topología árbol

En el diseño jerárquico, la planificación se vuelve más sólida al incorporar el concepto de enlaces troncales o "uplinks" para conectar las capas, este modelo permite analizar el enrutamiento entre switches y la habilidad del controlador para dirigir rutas óptimas a través de diferentes niveles de conmutación.

Tabla 8

Planificación IP y mapeo jerárquico - Escenario 2: Topología árbol

Dispositivo	Nodo de conexión en capa superior	Interfaz de enlace	Dirección IP lógica	Máscara de subred
Switch s1	N/A Capa core	N/A	N/A: capa 2	N/A
Switch s2	Switch s1: Capa de agregación izquierda	s2-eth3 (Uplink)	N/A: capa 2	N/A
Switch s3	Switch s1: Capa de agregación derecha	s3-eth3 (Uplink)	N/A: capa 2	N/A

Dispositivo	Nodo de conexión en capa superior	Interfaz de enlace	Dirección IP lógica	Máscara de subred
Host h1	Switch s2: Capa de acceso	h1-eth0	10.0.2.1	255.255.255.0
Host h2	Switch s2: Capa de acceso	h2-eth0	10.0.2.2	255.255.255.0
Host h3	Switch s3: Capa de acceso	h3-eth0	10.0.2.3	255.255.255.0
Host h4	Switch s3: Capa de acceso	h4-eth0	10.0.2.4	255.255.255.0

Nota. La segmentación de los hosts en el bloque 10.0.2.0/24 facilita la evaluación del enrutamiento entre switches manejado por el controlador. Fuente: Elaboración propia.

E. Jerarquía de encapsulamiento

El diseño final se basa en un modelo por capas que va desde la infraestructura física hasta las subredes lógicas separadas, esta estructura jerárquica permite a los estudiantes poder trabajar con una red que, aunque es virtual, sigue los mismos principios y estándares que una infraestructura de telecomunicaciones auténtica.

Abstracción de hardware y virtualización

La base de la arquitectura se fundamenta en el hardware físico (PC anfitrión), que ofrece los recursos necesarios de procesamiento, memoria y almacenamiento; en este nivel opera el hipervisor (VirtualBox), sirviendo como una capa de abstracción para gestionar la emulación de hardware y permitir la ejecución del sistema operativo invitado (Ubuntu).

Segmentación a través de Network Namespaces

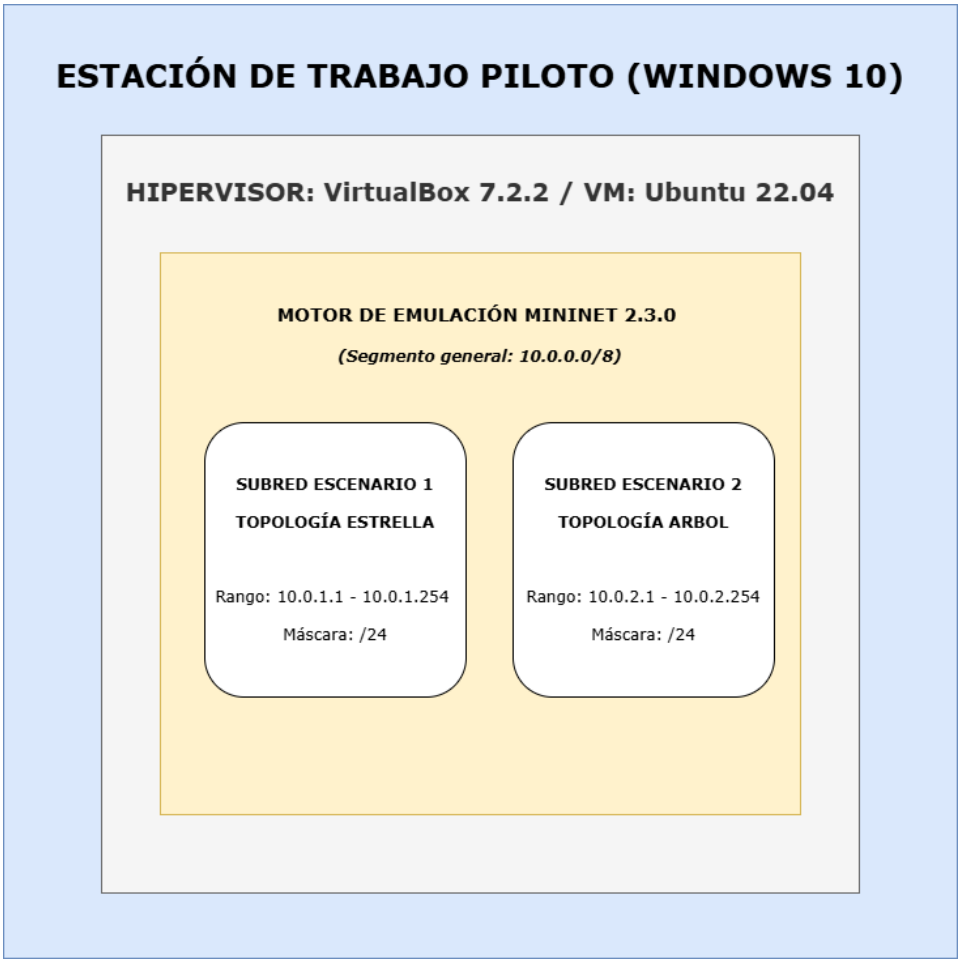
Dentro del sistema operativo Ubuntu, el motor de Mininet emplea la tecnología de Network Namespaces para crear los nodos de la topología, esta técnica posibilita una virtualización ligera donde cada host o switch emulado puede actuar como un proceso autónomo con su propia pila TCP/IP, tablas de rutas e interfaces dedicadas.

Organización de bloques anidados

Una estructura por capas garantiza que los parámetros de red establecidos funcionen con una alta precisión técnica, permitiendo que el laboratorio virtual sea un reflejo confiable y válido de los entornos físicos. Es por eso que se muestra el diagrama de bloques del direccionamiento lógico, la solución está estructurada en capas concéntricas para asegurar el aislamiento.

Figura 14

Diagrama de bloques del direccionamiento lógico y segmentación de red



Nota. El diagrama muestra la jerarquía de encapsulamiento desde el hardware físico hasta las subredes virtuales gestionadas por los espacios de nombres de red (Network Namespaces) de Mininet. *Fuente:* Autor.

La jerarquía se establece de la siguiente manera:

- **PC anfitrión:** Entorno físico esencial.
- **VirtualBox:** Capa de virtualización de tipo II.

- **Ubuntu:** Sistema operativo con el núcleo de Linux donde reside la lógica de red.
- **Mininet:** Plataforma de emulación que implementa los dispositivos virtuales.
- **Subredes aisladas:** Segmentos específicos (10.0.1.0/24 y 10.0.2.0/24) donde se llevan a cabo las prácticas experimentales de forma autónoma.

3.3 FASE 3 – IMPLEMENTACIÓN

Esta fase detalla el proceso técnico para establecer el laboratorio virtual en la estación de trabajo inicial, apoyándose en las especificaciones detectadas en la FASE 1. Siguiendo el paradigma de Infraestructura como Código (IaC), se lleva a cabo la conversión del diseño conceptual de la FASE 2 en un entorno operativo que es capaz de replicar las redes SDN como máxima precisión.

La ejecución se organiza de forma ordenada, comenzando con la configuración del entorno de virtualización, seguida de la programación de las topologías mediante scripts de Python y culmina con la orquestación del plano de control utilizando el controlador Ryu. Este método garantiza que el sistema pueda ser estable, funcional y fácilmente replicable en el laboratorio de telecomunicaciones de la UCSG.

3.3.1 Aprovisionamiento y configuración del hipervisor (VirtualBox)

El primer paso esencial en la implementación, es la configuración del hipervisor, que actuará como la "caja" que contendrá todo el ecosistema de red. Para este proyecto se ha seleccionado Oracle VM VirtualBox versión 7.2.2, un hipervisor de tipo 2 que facilita la abstracción de los recursos físicos de la estación piloto.

A diferencia de una instalación convencional, se opta por una configuración optimizada específica para la virtualización de redes, ajustando la distribución de CPU y RAM conforme a las capacidades detectadas en el procesador Intel Core i5-5200U. Además, se prioriza la activación de tecnologías de aceleración de hardware (VT-x) para disminuir la latencia en el intercambio de datos y se gestiona un almacenamiento dinámico para maximizar el uso de los 375 GB disponibles en la estación.

Figura 15

Versión del hipervisor Oracle VM VirtualBox

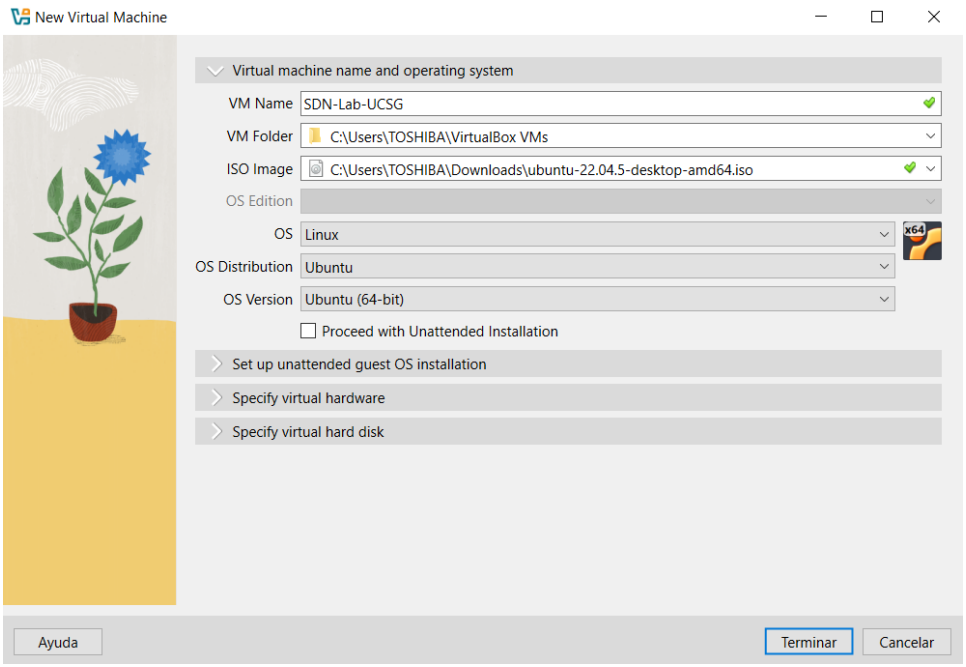
Nota. Información detallada de la versión de Oracle VM VirtualBox instalada en el equipo anfitrión. Este software actúa como el hipervisor encargado de gestionar y virtualizar el entorno de pruebas para la red SDN. *Fuente*: Autor.

A. Creación de la instancia virtual y selección de arquitectura

La creación del entorno comienza con la generación de la instancia virtual, que se ha nombrado técnicamente como “SDN_Lab_UCSG”. La exactitud en este paso es muy importante para de esta manera asegurar que el sistema operativo invitado pueda interactuar adecuadamente con los recursos del anfitrión.

Figura 16

Asistente de configuración inicial de la máquina virtual



Nota. La captura de pantalla detalla los parámetros de creación de la máquina virtual "SDN-Lab-UCSG" en VirtualBox. *Fuente:* Autor.

Razonamiento detrás de la arquitectura de 64 bits

Se ha optado por una arquitectura de 64 bits para la máquina virtual, esta elección es importante para poder garantizar la completa compatibilidad con el núcleo de Linux (Ubuntu 22.04), que necesita instrucciones de 64 bits para manejar de forma eficiente los Network Namespaces y los grupos de control (cgroups) que utiliza Mininet para la emulación de nodos.

Selección del tipo y versión

Durante el asistente de configuración, se establece el tipo de sistema como: Linux y la versión como Ubuntu (64 bit desktop); esta configuración permite que VirtualBox preconfigure las extensiones de seguridad y las interfaces de red virtuales necesarias para la implementación del plano de datos.

Compatibilidad de kernel

Al elegir esta arquitectura se asegura que el controlador Ryu, que utiliza Python 3, pueda ejecutarse y funcione en un entorno de memoria ampliada o extendida, de

esta manera evita las congestiones también llamadas cuellos de botella, durante el procesamiento de tablas de flujo complejas.

B. Gestión y asignación de recursos de procesamiento (vCPU) y memoria RAM.

Una vez que se ha establecido la estructura, el siguiente paso a completar es la configuración del hipervisor, este consiste en definir que recursos de hardware del host se van a ceder a la instancia virtual; esta asignación requiere un enfoque equilibrado para asegurar que el rendimiento en ambos sistemas no se vea afectado.

Distribución de memoria RAM

Se ha decidido asignar un total de 4096 MB (4GB) a la máquina virtual, esta cifra corresponde al 50% de la memoria física disponible en la estación piloto, la cual contiene 8 GB de memoria RAM, de esta manera, garantiza que el sistema operativo Ubuntu, pueda tener suficiente capacidad para iniciar el controlador Ryu y herramientas de monitoreo como Wireshark, sin la necesidad de depender del uso de memoria de intercambio, lo que podría aumentar la latencia de la red.

Configuración de procesamiento (vCPU)

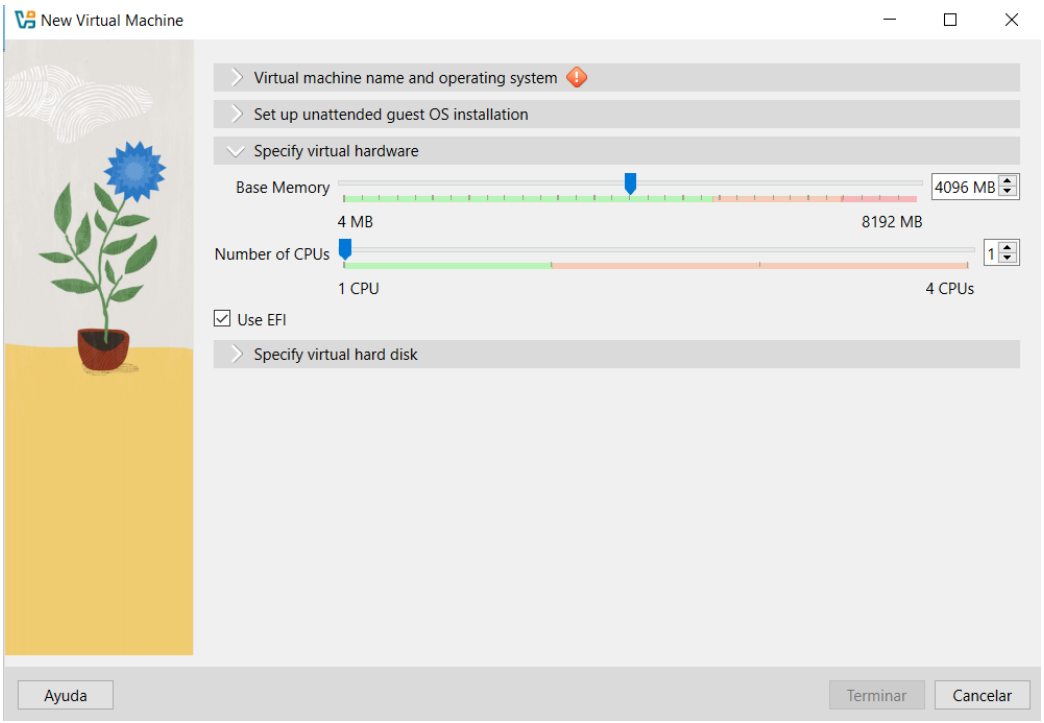
En el entorno de emulación se asigna un solo procesador virtual (vCPU), considerando que el procesador Intel Core i5-5200U correspondiente a la estación de trabajo piloto, cuenta con 2 núcleos físicos y 4 hilos lógicos, mantener una vCPU dedicada, permite que Mininet gestione las interrupciones de red de manera aislada, mientras los recursos restantes del host preservan la estabilidad de VirtualBox y del sistema operativo Windows que se tiene como base.

Optimización del límite de ejecución

Se establece el límite de ejecución al 100% para la vCPU asignada, asegurando así que el motor de conmutación Open vSwitch pueda manejar picos de tráfico elevados durante las pruebas de rendimiento con iperf, sin restricciones artificiales que son impuestas por el hipervisor.

Figura 17

Asignación de recursos de hardware para la máquina virtual



Nota. La captura muestra el interfaz de especificación de hardware donde se definen los recursos del sistema para la máquina virtual, asignando 4096 MB de memoria y un procesador. *Fuente*: Autor.

C. Configuración de aceleración de hardware y virtualización anidada

La efectividad de una red emulada por software está directamente relacionada con la habilidad del hipervisor para aprovechar las funciones avanzadas del procesador físico; en esta sección se describe la configuración de las tecnologías de virtualización que permiten que la estación piloto i5-5200U maneje el tráfico de red con la menor latencia posible.

Activación de Intel VT-x y paginación anidada

Se ha confirmado la activación de las extensiones de virtualización Intel VT-x, esta innovación permite que el sistema operativo invitado Ubuntu, pueda ejecutar instrucciones directamente con el procesador físico, eliminando la necesidad de emular cada acción a través de software, lo cual es clave para que el switch virtual Open vSwitch, pueda procesar paquetes en microsegundos. Además, se activa la paginación anidada (EPT), la cual mejora la gestión de la memoria RAM entre el host y la máquina

virtual, de esta manera disminuye la carga del sistema durante pruebas con tráfico intensivo.

Interfaz de Paravirtualización KVM

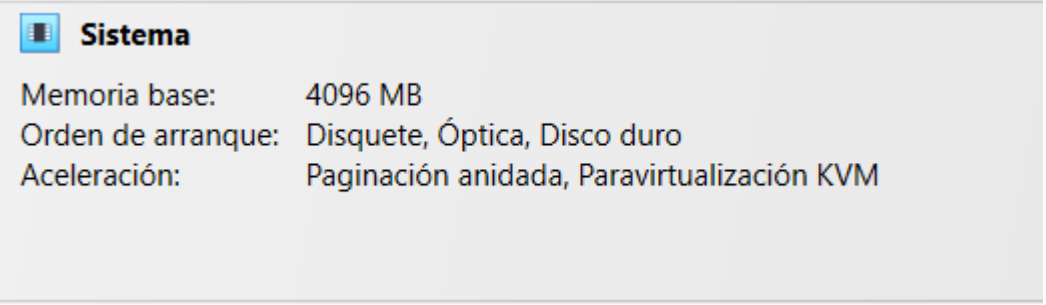
Se ha elegido la interfaz de paravirtualización KVM (por sus siglas en inglés Kernel-based Virtual Machine) en la configuración de VirtualBox, esto le indica a Ubuntu que opera bajo un hipervisor compatible, lo que permite que el núcleo de Linux ajuste sus funciones de red para colaborar con VirtualBox, resultando en una emulación de hosts y switches mucho más fluida y muy cercana al comportamiento de un hardware auténtico.

Importancia en el entorno SDN

Sin estas optimizaciones la comunicación de mensajes de OpenFlow entre el conmutador y el controlador Ryu, tendrían retardos artificiales que comprometerían las evaluaciones de eficiencia; al proporcionar acceso directo al hardware, se garantiza que los datos de latencia obtenidos en las pruebas que se realizarán, representen con precisión el potencial de las redes definidas por software.

Figura 18

Resumen de los parámetros del sistema de la máquina virtual



Nota. La figura muestra que en la VM se encuentra activa la característica de aceleración por hardware (página anidada y paravirtualización de KVM) diseñadas para optimizar el rendimiento de la virtualización. *Fuente:* Autor.

D. Definición del subsistema de almacenamiento dinámico

La gestión del almacenamiento es un factor importante para la portabilidad y el rendimiento del laboratorio virtual, en este subtema se explica la configuración del disco duro virtual, priorizando un enfoque que optimice el espacio disponible en el disco físico de la estación de trabajo piloto.

Uso de imágenes de disco VDI

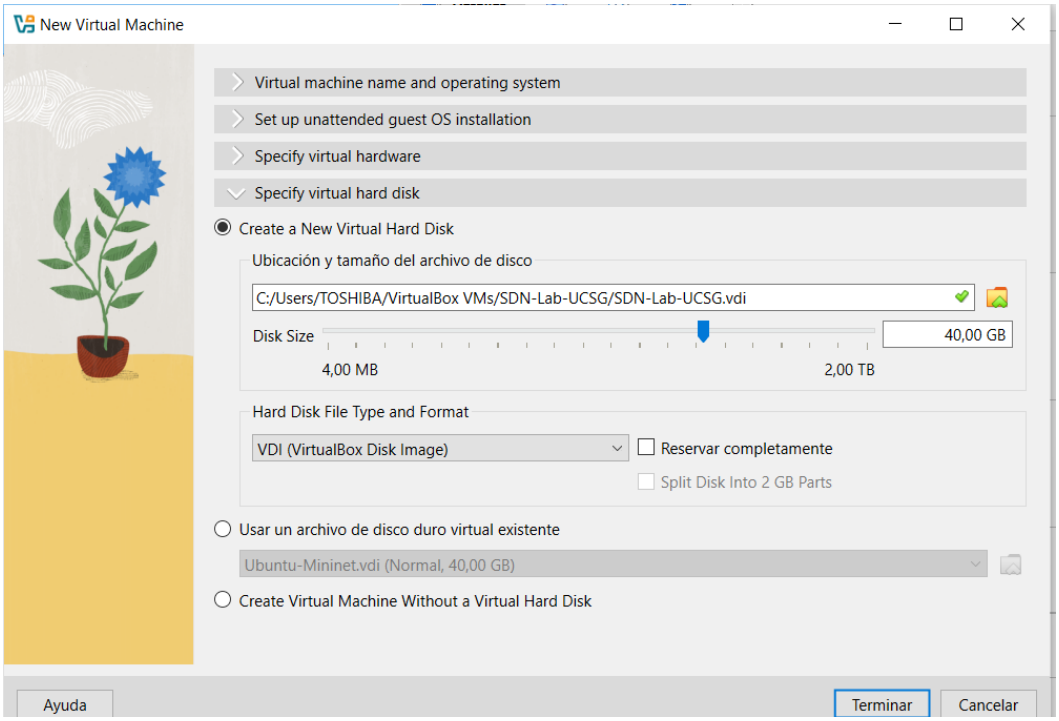
Para la máquina virtual se ha optado por el formato VDI (VirtualBox Disk Image), que es el estándar nativo del hipervisor, este formato facilita una integración sin problemas con las operaciones de entrada y salida del sistema anfitrión, asegurando que la instalación de bibliotecas de Mininet y el controlador Ryu se lleve a cabo de forma completa y sin errores.

Dimensionamiento de la unidad

Se ha establecido un límite máximo de 40 GB para el disco virtual, esta capacidad es adecuada para alojar el sistema operativo Ubuntu, el entorno de emulación SDN y las capturas de tráfico de Wireshark que se generarán más adelante en las pruebas a realizar, asegurando que los estudiantes no se vean limitados por el espacio durante la ejecución de las prácticas.

Figura 19

Configuración del disco duro de la máquina virtual



Nota. La captura muestra la ventana de configuración que ilustra la creación de un disco duro virtual nuevo de 40 GB tipo VDI, especificando la ruta donde se guardará el archivo del disco para el laboratorio SDN. *Fuente:* Autor

Almacenamiento de expansión dinámica

El disco virtual ha sido configurado empleando el principio de almacenamiento dinámico, esta elección técnica es fundamental para la efectividad, ya que permite que el archivo de la máquina virtual ocupe inicialmente solo un mínimo espacio físico de los 375 GB disponibles en el disco, aumentando únicamente a medida que se incorporen datos y scripts al sistema operativo Ubuntu. Esto previene el derroche de recursos de almacenamiento en la estación piloto, permitiendo que el espacio restante sea aprovechado por el sistema Windows para otras funciones administrativas de la investigación.

E. Configuración inicial de adaptadores de red para el entorno SDN

La conectividad de la máquina virtual es el elemento que establece las llamadas "tuberías" lógicas a través de las cuales transitarán tanto el tráfico de gestión como el tráfico de datos emulado, para asegurar un entorno operativo, se ha elaborado un diseño de red que posibilita la interacción con recursos externos y la visibilidad del tráfico desde la estación piloto.

Interfaz de acceso a internet - Modo NAT

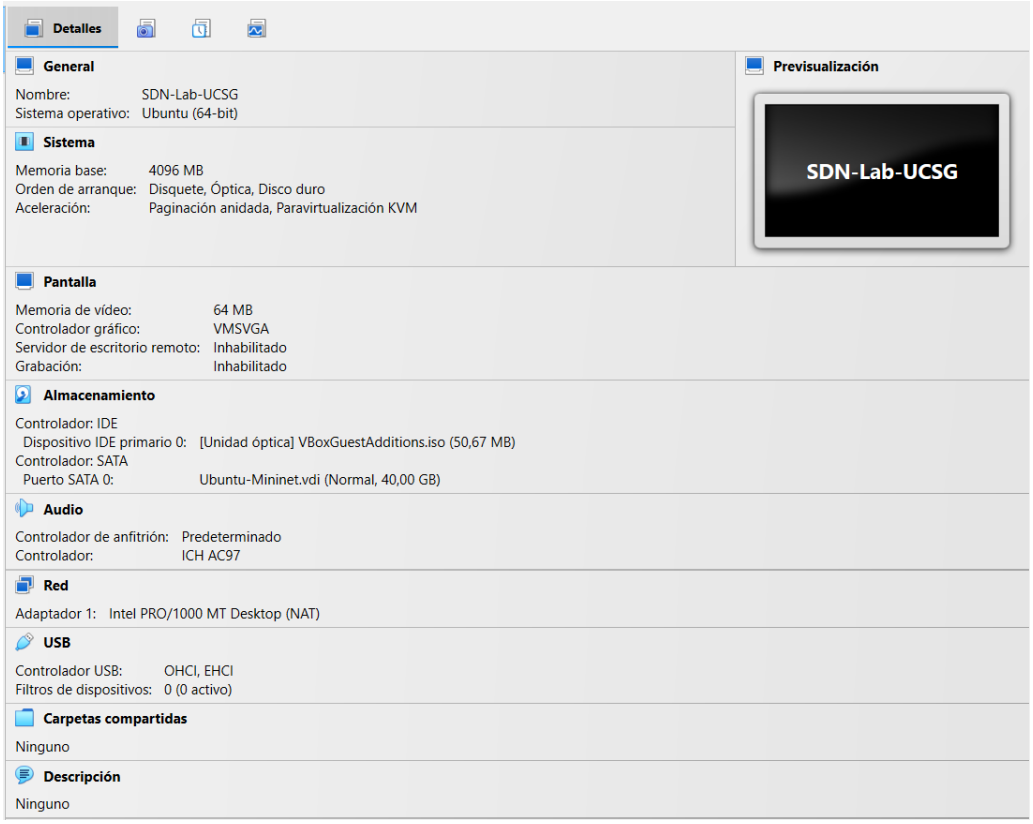
El primer adaptador de red se ha configurado en modo NAT (Network Address Translation), esta opción permite que la instancia virtual SDN-Lab-UCSG aproveche la conexión a internet de la estación piloto para llevar a cabo las actualizaciones esenciales de repositorios y la descarga de las bibliotecas requeridas para el controlador Ryu y Mininet.

Interfaz de gestión y análisis - Modo anfitrión

Se activa un segundo adaptador en modo anfitrión (Host-Only) o como un adaptador puente, según la necesidad de visibilidad del exterior, el propósito de esta interfaz es permitir una comunicación directa entre el sistema Windows 10 y la máquina virtual, facilitando que herramientas de monitoreo externas puedan examinar el plano de control SDN sin las limitaciones de seguridad que impone una red NAT cerrada.

Figura 20

Resumen general de las especificaciones de la máquina virtual



Nota. Panel principal de la máquina virtual que muestra la vista general de todos los recursos asignados, incluyendo detalles de pantalla, almacenamiento, audio y el adaptador de red configurado. *Fuente:* Autor.

Importancia para el monitoreo con Wireshark

La implementación de estos adaptadores es bastante fundamental para alcanzar el objetivo específico de análisis de tráfico, al separar el tráfico de internet del tráfico de datos interno, los estudiantes pueden filtrar eficazmente las tramas OpenFlow 1.3 en Wireshark, minimizando las interferencias causadas por paquetes del sistema o actualizaciones de fondo.

3.3.2 Instalación y optimización de Ubuntu 22.04.5 LTS

Una vez que la arquitectura de virtualización ha sido establecida, el siguiente reto técnico es equipar el entorno con un sistema operativo que funcione como motor de red, en este punto no se trata simplemente de llevar a cabo una instalación convencional, sino de preparar el entorno para que herramientas exigentes como

Mininet y Ryu funcionen sin problemas dentro de las capacidades del procesador Intel Core i5-5200U.

El rendimiento del sistema invitado es fundamental en esta etapa, una configuración liviana y optimizada permite que los recursos de la estación piloto se destinen a la conmutación de paquetes en lugar de procesos en segundo plano innecesarios. A continuación, se describe el procedimiento desde el arranque inicial hasta la depuración del sistema destinado a un entorno SDN de alta calidad.

A. Montaje de la imagen ISO y secuencia de arranque inicial

El proceso de la implementación del sistema operativo comienza con la conexión del medio digital de instalación a la infraestructura virtual, este paso es importante para garantizar que el ciclo de vida del sistema comience con los requerimientos de 64 bits acordes a las especificaciones del ecosistema SDN.

Selección y reconocimiento del medio

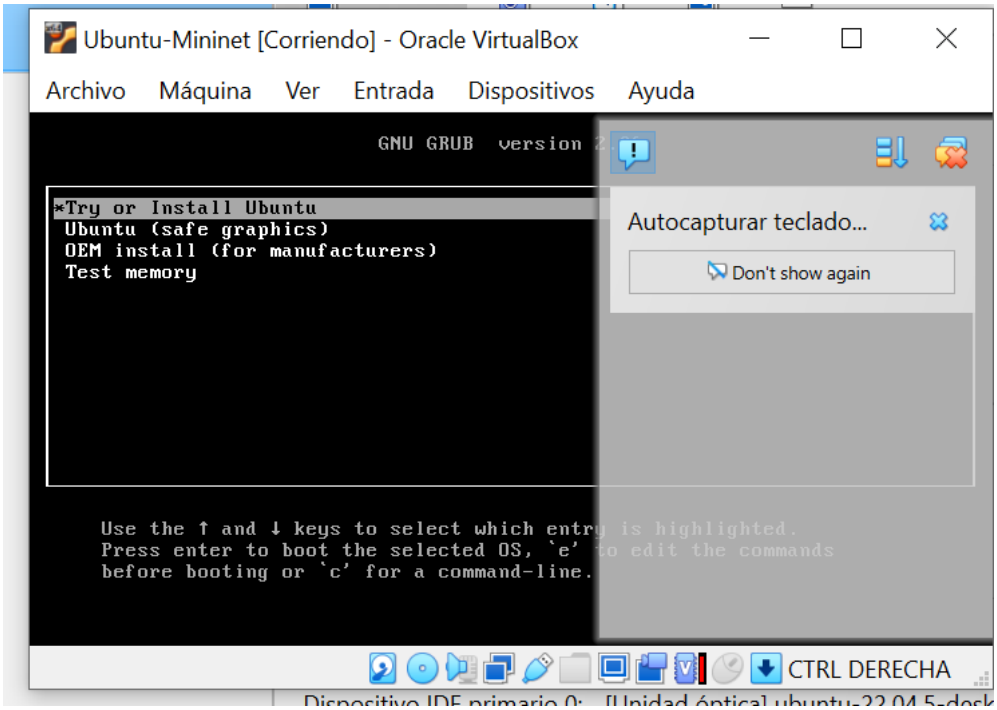
Se lleva a cabo la descarga de verificación de la imagen oficial de Ubuntu 22.04.5 LTS (Jammy Jellyfish) en su versión de 64 bits. Es esencial que el hipervisor reconozca el archivo .iso como una unidad virtual de CD/DVD, permitiendo que el kernel de Linux acceda a los módulos de instalación de forma nativa sin errores de lectura que podrían afectar la estabilidad futura de Mininet.

Priorización en la BIOS virtual

En los ajustes de "Sistema", se cambia el orden de arranque, estableciendo la unidad óptica como el primer dispositivo del inicio; esta configuración es vital para asegurar que el instalador se cargue antes que el disco duro virtual (VDI), que inicialmente está vacío, garantizando un arranque adecuado hacia el entorno de instalación.

Figura 21

Interfaz de selección del GNU GRUB para Ubuntu 22.04.5 LTS en la VM



Nota. En la captura se muestra el GNU GRUB, el gestor de arranque predeterminado de casi todas las distribuciones de Linux, este es el primer programa que se ejecuta cuando la computadora termina de cargar la configuración de la BIOS. *Fuente:* Autor

Luego, cuando el hipervisor ya ha priorizado la unidad óptica, el cargador de arranque inicial se despliega, después de esta etapa, el sistema espera la confirmación del usuario para comenzar el entorno de despliegue en la memoria RAM, antes de tocar el disco duro físico virtual.

Carga del instalador "Ubiquity"

Al iniciar la instancia virtual, el sistema activa el asistente de instalación gráfico conocido como Ubiquity, este proceso se lleva a cabo con una verificación de la integridad del medio, asegurando que todos los archivos comprimidos del sistema, se puedan desplegar de manera correcta en la unidad de 40 GB previamente reservada en la estación piloto.

B. Definición del particionamiento lógico y perfil de usuario

Después de completar la fase de arranque inicial desde el menú de GRUB, donde se selecciona la opción "Try or Install Ubuntu" para dar inicio al asistente

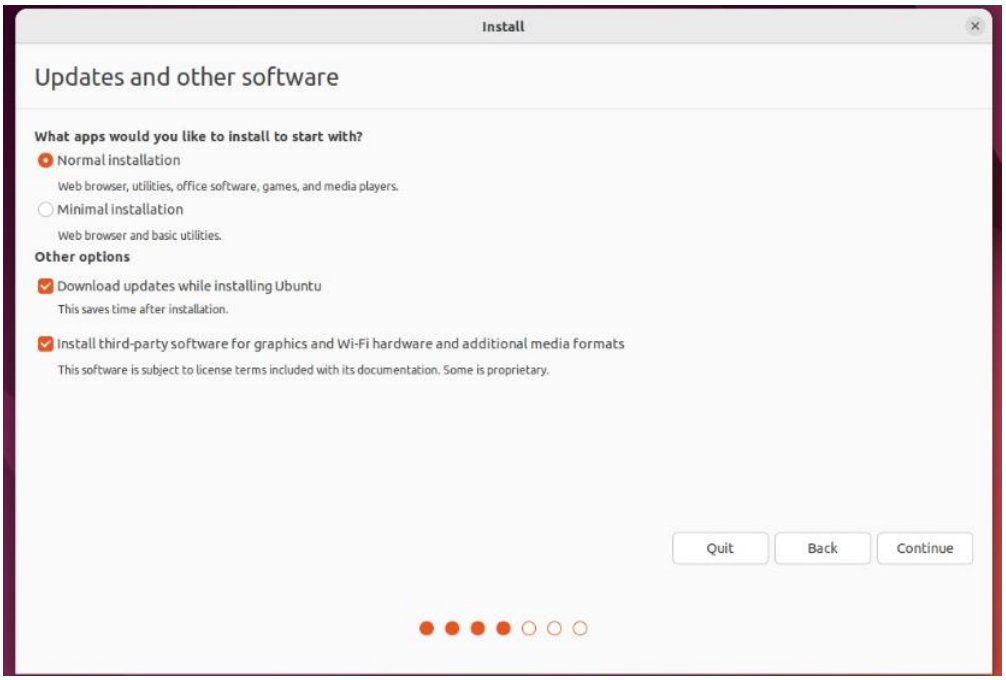
gráfico, se avanza con la configuración interna del sistema. Este paso es muy importante ya que las elecciones realizadas en el instalador determinan la eficiencia con el que el procesador i5-5200U manejará las herramientas de red posteriormente,

Elección de software y actualizaciones en tiempo real

Durante esta etapa se escoge la opción de "Instalación normal" para garantizar la disponibilidad de herramientas básicas y navegadores que faciliten la descarga de dependencias adicionales. Es muy importante activar las opciones "Descargar actualizaciones mientras se instala Ubuntu e "Instalar software de terceros para gráficos", tal como se evidencia a continuación:

Figura 22

Configuración de actualizaciones durante la instalación



Nota. Esta configuración permite que el sistema obtenga parches de seguridad y controladores necesarios desde el inicio, asegurando la estabilidad del núcleo de Linux y su compatibilidad con las funciones de aceleración de hardware en la estación piloto. *Fuente:* Autor.

Estructura de datos y gestión de volúmenes (LVM)

En el apartado del tipo de instalación, se elige el borrado total del disco virtual de 40 GB utilizando el sistema de LVM (Gestión de Volúmenes Lógicos). La

implementación de LVM es una decisión técnica estratégica para el laboratorio de la UCSG; a diferencia de las particiones tradicionales, los volúmenes lógicos permiten una adaptación dinámica del espacio de almacenamiento, esto asegura que, si los archivos de captura de tráfico en formato .pcap de Wireshark aumentan significativamente, el sistema puede ampliar su capacidad sin riesgo de pérdida de información o necesidad de formateo.

Identidad del entorno de red

Finalmente, se establece el perfil de usuario y el nombre de la instancia, que se registra como “SDN-Lab-UCSG”, este nombre busca facilitar una identificación clara en el administrador de VirtualBox y establece la base para la estructura de directorios donde se almacenarán los scripts de topología lab_ucsg.py; la creación de una cuenta con privilegios administrativos (sudo) es esencial, ya que la manipulación de interfaces de red virtuales y la ejecución del emulador Mininet requieren acceso a funciones restringidas del sistema operativo.

C. Actualización crítica de repositorios y parches de seguridad

A pesar de que el proceso de instalación inicial incluyó la descarga de paquetes fundamentales, es vital llevar a cabo una actualización manual y completa una vez que el sistema operativo está en funcionamiento en el escritorio. Este paso asegura que la base de datos de los repositorios locales este completamente sincronizada con los servidores oficiales de Ubuntu, garantizando que las dependencias de Red para mini net y Ryu se instalen sin conflictos de versiones.

Sincronización del núcleo para funciones SDN

La simulación de nodos dentro de un entorno virtual requiere de módulos específicos del núcleo de Linux, como los Network Namespaces y los cgroups, que posibilitan el aislamiento del tráfico de cada host virtual. Dado que estos elementos son objetos de constantes actualizaciones, llevar a cabo mejoras críticas garantiza que la estación i5-5200U maneje estas operaciones con la máxima estabilidad, reduciendo al mínimo los riesgos de errores por desbordamiento de memoria o fallos en el plano de datos.

Mantenimiento mediante línea de comandos a través de la terminal

Este proceso se efectúa mediante la terminal de Ubuntu, utilizando el comando "sudo apt update" para actualizar los índices de paquetes y "sudo apt upgrade" para implementar las actualizaciones de seguridad acumuladas.

El proceso se complementa con el comando "sudo apt autoremove", que desinstala paquetes innecesarios y dependencias sobrantes del ciclo de instalación inicial. Esta limpieza es crucial para mantener la imagen virtual ligera y eficiente, asegurando que los 2 GB de RAM se destinen prioritariamente al procesamiento de protocolos OpenFlow en lugar de a tareas de sistema en segundo plano.

Verificación de la estabilidad del entorno

Al concluir este ciclo de actualización, el sistema operativo pasa de ser una instalación estándar a convertirse en una plataforma de red sólida y verificable. La verificación de la ausencia de actualizaciones pendientes funciona como el último control, permitiendo que el despliegue del software SDN en etapas posteriores se realice sobre una base tecnológica fiable.

D. Implementación de Guest additions y unión de hardware

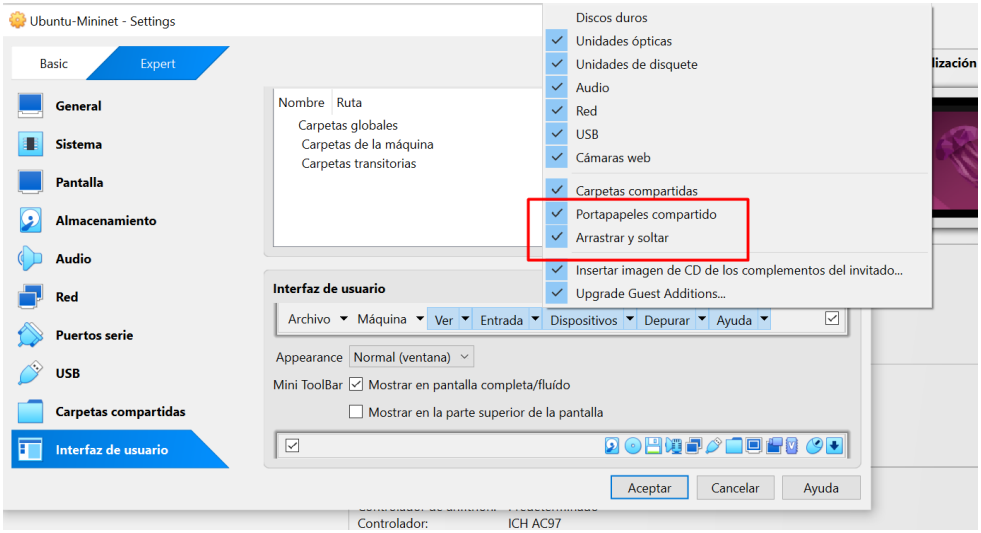
Una vez actualizado el sistema operativo, el siguiente paso fundamental es llevar a cabo la instalación de las guest additions de VirtualBox. Este conjunto de controladores y programas es esencial para una conexión fluida entre la estación de trabajo piloto (Windows 10) y el entorno de pruebas (Ubuntu), mejorando la utilización del procesador i5-5200U.

Sincronización y optimización de recursos

La incorporación de estos complementos es crucial para que el procesador Intel core i5-5200U administre las interrupciones del teclado y ratón de forma efectiva, previniendo retrasos en la interfaz gráfica. Además, proporciona características de productividad como el portapapeles compartido y la función de arrastrar y soltar como herramientas necesarias para transferir secciones de código desde el anfitrión hasta la terminal de Ubuntu, disminuyendo el riesgo de errores humanos en la programación.

Figura 23

Habilitación de "portapapeles compartido" y "arrastrar y soltar"



Nota. Menú de opciones de VirtualBox donde se activan las herramientas de integración para compartir el portapapeles y permitir el arrastre de archivos bidireccional, mejorando la usabilidad durante la configuración de la red. *Fuente:* Autor.

Mejora gráfica para monitoreo

Al instalar los controladores de video de VirtualBox, se activa la aceleración gráfica en 2D y 3D coma permitiendo que aplicaciones de análisis de tráfico como Wireshark funcionen sin problemas. Sin esa integración como la visualización de gráficos de tráfico en tiempo real podría agotar los ciclos de la CPU, afectando el rendimiento de la conmutación de paquetes SDN.

Proceso de instalación

Para realizar esta tarea, se monta la imagen de CD virtual de la guest additions desde el menú de VirtualBox y se ejecuta el script de instalación usando la terminal con el comando “sudo ./VBoxLinuxdditions.run”. Es esencial reiniciar la instancia para que los modelos del núcleo se carguen correctamente, lo que asegura que el sistema reconozca dinámicamente la resolución de la pantalla y mejore la administración de las carpetas compartidas donde se guardarán los resultados de las prácticas.

E. Ajustes de la terminal y disponibilidad del entorno

Como último paso antes de la implementación del entorno SDN, se procede a la optimización del espacio de trabajo y configuración de la terminal de comandos. Dado que la gestión de Mininet y Ryu depende únicamente de texto, contar con una terminal bien ajustada es muy importante para las prácticas de los estudiantes.

Optimización de servicios y ahorro de memoria RAM

Para maximizar los recursos de la estación de trabajo piloto, que es el computador anfitrión, se procede a desactivar aplicaciones innecesarias que se activan automáticamente en Ubuntu desktop, como las actualizaciones automáticas en segundo plano y los efectos visuales pesados. Esta limpieza técnica garantiza que la mayor parte de los 2 GB o 4 GB de RAM disponibles permanezcan accesibles para el controlador Ryu y la base de datos de flujos de Open vSwitch, asegurando la estabilidad del entorno de trabajo durante las pruebas de carga realizadas con iperf.

Configuración de la pila de ejecución

Se comprueba que Python 3 y su gestor de paquetes "pip" estén correctamente instalados, dado que el controlador Ryu depende de este lenguaje. Las variables de entorno se modifican en el archivo ".bashrc" para que los ejecutables de Mininet sean accesibles globalmente, facilitando así que los estudiantes comiencen el laboratorio desde cualquier carpeta en la terminal y mejorando su experiencia educativa.

Estado de disponibilidad final

Gracias a estas configuraciones, el sistema operativo se transforma de una instalación estándar a una estación de trabajo SDN mejorada. Se verifica la conectividad utilizando un comando sencillo de red y se garantiza que el acceso a través de la terminal sea rápido, preparando el terreno de forma integral para la instalación de software de emulación y control en la próxima fase.

3.3.3 Instalación del ecosistema SDN (Mininet – Ryu – OVS)

En este punto se despliegan las herramientas de software que conforman la arquitectura de la red SDN del laboratorio, a diferencia de una instalación normal, aquí se prioriza la configuración de dependencias y la optimización del entorno de ejecución, todo con el fin de asegurar que el controlador basado en Python pueda orquestar los dispositivos de red virtuales sin latencia en la estación de trabajo piloto.

El enfoque principal es transformar la instancia de Ubuntu, en un ecosistema programable en el que el plano de datos y el de control puedan interactuar bajo los estándares del protocolo OpenFlow 1.3, esto garantiza que la emulación sea de alta fidelidad en las prácticas a realizar.

A. Implementación del plano de datos: Mininet y Open vSwitch (OVS)

La creación del plano de datos forma la base operativa de la red, donde se especifican los switches y hosts encargados de manejar el tráfico. En este caso, se eligió: Mininet como el emulador y Open vSwitch para actuar como el switch virtual responsable de la conmutación de datos.

Integración directa con el núcleo de Linux

En lugar de utilizar compilaciones externas, se ha optado por los repositorios oficiales de Ubuntu 22.04 LTS para asegurar versiones estables de estas herramientas. Esta elección es estratégica, porque garantiza que Mininet se enlace de forma nativa con los módulos de red del núcleo, permitiendo que la configuración de interfaces virtuales y puentes de red, se realice con baja carga de CPU en el procesador i5-5200U, lo cual es vital para mantener la precisión en las cifras de rendimiento.

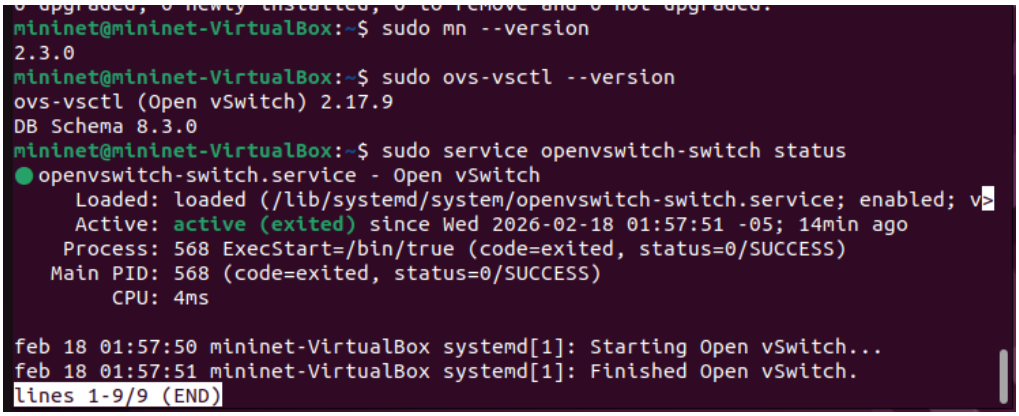
Proceso de instalación y pruebas de conectividad

El proceso comienza ejecutando el comando “sudo apt install mininet -y”, que instala el emulador junto con las bibliotecas requeridas para el funcionamiento de Open vSwitch, durante esta instalación, es opcional concluir con el paquete “openvswitch-testcontroller”, una herramienta ligera que permite realizar verificaciones rápidas de conectividad (ping all), antes de pasar el control total al controlador Ryu.

Luego, se ejecutan los comandos de diagnóstico con el fin de obtener la versión exacta del componente; esto es importante ya que al conocer las versiones que estamos utilizando, prevenimos el error de dependencia, donde pueden ocurrir problemas futuros si se usa una versión antigua, no compatible con otra dependencia, para este caso: bajo el comando “sudo mn –version” se puede verificar que se instaló correctamente la versión 2.3.0 de Mininet, como se muestra a continuación:

Figura 24

Verificación de instalación de Mininet y Open vSwitch



Nota. Captura de la terminal de Ubuntu mostrando la ejecución de comandos para consultar la versión de Mininet y Open vSwitch. Se visualizan las respectivas versiones y se confirma estado activo y sin errores. *Fuente:* Autor.

Administración de OVS cómo servicio del sistema

Un elemento clave de esta configuración es asegurar que Open vSwitch funcione como un servicio persistente a través de “systemctl”; al funcionar como un servicio en segundo plano, se asegura que los puertos virtuales y las fuentes de red estén listos desde el momento en que Mininet comienza a crear la topología, evitando problemas de sincronización entre el hardware virtualizado y el software de administración.

B. Implementación del plano de control: Ryu SDN Framework

Después de establecer el plano de datos, se continúa con la instalación del plano de control, representado por el “framework Ryu”. Esta herramienta es un componente de software que ofrece una arquitectura de control abierta, permitiendo que la gestión de la red se realice de forma programática y centralizada mediante el protocolo OpenFlow.

Administración de dependencias y entorno de Python

A diferencia de los controladores tradicionales, Ryu está completamente construido en Python 3, lo que requiere un entorno de ejecución moderno y efectivo en la estación piloto. El proceso de implementación inicia con la instalación de herramientas esenciales como “python3-pip” y “python3-dev”. La inclusión de

bibliotecas de desarrollo (-dev) es importante, ya que habilita la creación de extensiones de bajo nivel necesarias para que el controlador procese datos rápidamente en el procesador i5-5200U, asegurando que no haya cuellos de botella durante la gestión de los flujos de red.

Implementación a través del gestor de paquetes "pip"

La instalación del marco se lleva a cabo empleando el comando "pip3 install ryu", lo cual asegura que se obtenga la versión más estable y apta para Ubuntu 22.04. Este enfoque de implementación garantiza que todas las bibliotecas de sus áreas (como Eventlet para la gestión de hilos y Msgpack para la serialización de datos) se sincronicen adecuadamente, permitiendo que el controlador Ryu administre múltiples switches virtuales al mismo tiempo sin experimentar pérdida de paquetes de control.

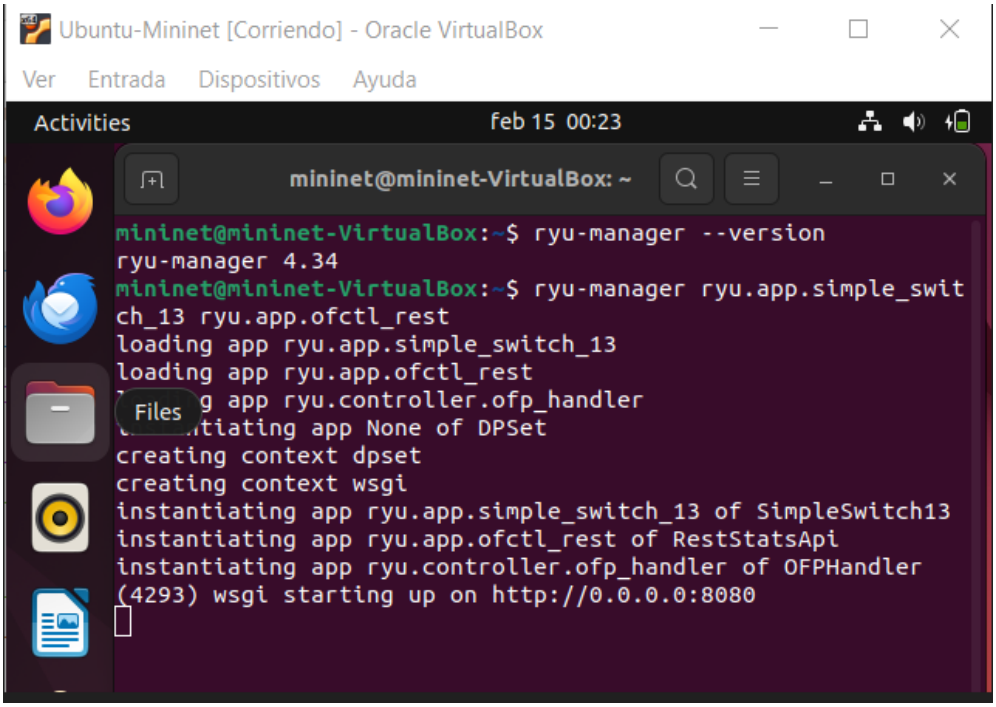
Configuración de variables de entorno y accesibilidad

Con el fin de mejorar la experiencia en el laboratorio, se ajusta el sistema para que el comando "ryu-manager" sea reconocido de manera global en la terminal de Ubuntu. Esto se efectúa modificando las variables de entorno en el perfil del usuario, lo que permite a los estudiantes poder lanzar cualquier aplicación del control (como el switch de aprendizaje o el monitor de tráfico) desde cualquier ubicación, facilitando la ejecución de las prácticas en el laboratorio de telecomunicaciones.

Después de que se ejecutaron los respectivos comandos para agregar Ryu al "Path" y este sea reconocido, se procede a ejecutar el comando "ryu-manager –version", posterior se verifica la versión instalada de Ryu y que se mantenga de manera funcional, para finalmente, iniciar el servicio. Como se muestra en la Figura 25, se ejecutó el comando "ryu-manager ryu.app.simple_switch_13 ryu.app.ofctl_rest" con la finalidad de que se ejecute el controlador y empiece a actuar como el cerebro de la operación.

Figura 25

Verificación de la versión de Ryu y ejecución del controlador



Nota. Captura de la terminal de Ubuntu donde se verifica la versión instalada de Ryu y se inicia el servicio. Se muestra la carga de los módulos de switch simple y la API REST. *Fuente:* Autor.

Poder visualizar estos identificadores de versión en la pantalla de la terminal de Ubuntu no se trata solo de un trámite de verificación, más bien, es la evidencia de que el entorno virtual que se está creando, cuenta con una base de software coherente, esto garantiza que cuando los estudiantes procedan con el levantamiento de topologías personalizadas, los mensajes de control se procesarán con la latencia mínima que se espera de las redes SDN.

C. Orquestación y verificación del demonio “ovs-vswitchd”

Una vez que se han instalado los paquetes de software, la atención se centra en los procesos que se ejecutan en segundo plano del sistema operativo, el componente más importante en esta etapa es el demonio “ovs-vswitchd”, qué forma el "músculo" de la operación del laboratorio. Aunque mi niña es responsable de crear la ilusión de múltiples dispositivos, es este proceso el que realiza la conmutación y gestión de las tablas de flujo que el controlador envía posteriormente.

Verificación mediante administración de sistemas

Para garantizar que la infraestructura sea estable se utiliza la herramienta systemctl para verificar el estado del servicio de switch openvswitch, esta verificación es obligatoria antes de intentar crear una topología porque si el demonio no funciona los conmutadores virtuales no tendrán forma de procesar tramas, lo que convertirá a la red en un conjunto aislado de no sin comunicación.

Interpretación técnica de la terminal

Al realizar diagnósticos de estado, el propósito es confirmar que el servicio se encuentra en estado "activo (en ejecución)", una salida exitosa de la terminal no solo indica que el programa está encendido, sino que también confirma que el conmutador virtual ha cargado los módulos del kernel de Linux. Esta integración con el kernel permite que el tráfico fluya con baja latencia en el banco de pruebas utilizando las optimizaciones que hicimos anteriormente en la configuración de la máquina virtual.

Acceso a módulos de red

La orquestación adecuada de este demonio garantiza que Open vSwitch tenga permisos para manipular las interfaces de red virtuales que MiOninet va a crear para representar conexiones entre host y conmutadores. Sin esta validación de integridad, el laboratorio experimentaría fallas de conexión intermitentes, lo que dificultaría la obtención de datos precisos para las pruebas del rendimiento.

D. OpenFlow 1.3 – Configuración del protocolo de comunicación

Una vez que los servicios del sistema estén operativos, el siguiente paso importante es estandarizar el lenguaje de comunicación entre los planos de la red. No basta con permitir una conexión general; para lograr los objetivos académicos de esta práctica de laboratorio, es obligatorio utilizar el protocolo OpenFlow versión 1.3., esta decisión es importante porque garantiza que la infraestructura no se va a degradar a versiones obsoletas, tal es el caso de la versión 1.0, ya que esto limita significativamente la capacidad de los estudiantes para poder aprender y experimentar.

Versión 1.3 - justificación técnica

La selección de Open flow 1.3 sobre versiones anteriores se basa principalmente en la necesidad de introducir funciones de red avanzadas para las prácticas de redes LAN, A diferencia de la versión 1.0, la versión 1.3 introduce

características críticas como el manejo de tablas múltiples, grupos para la replicación de tráfico y contadores para el control de ancho de banda y calidad de servicio punto basado en las pautas de la ONF (Open Networking Foundation), esta versión representa el equilibrio óptimo entre estabilidad y riqueza de funciones para entornos de investigación y educación superior.

Protocolos de configuración y "forzado"

Para garantizar que Open vSwitch solo funciona según este estándar, se deben definir parámetros específicos en Mininet durante el inicio de la topología, técnicamente esto se logra configurando los protocolos en el conmutador virtual mediante comandos como: "ovs-vsctl set bridge [nombre_switch] protocols=OpenFlow13.".

Esta opción evita que el conmutador intente negociar versiones inferiores, lo que obliga a todos los intercambios de mensajes (Packet-In, Flow-Mod, Stats) a adherirse estrictamente a las estructuras de datos de la versión 1.3.

Compatibilidad con Ryu

Dado que el controlador ryu tiene soporte integrado y sólido para OpenFlow 1.3, esta configuración evita errores de protocolo de enlace que ocurren a menudo cuando hay una discrepancia de versión entre planos. La estandarización del canal de comunicación garantiza que el procesador Intel Core i5-5200u puede manejar las reglas de flujo de manera eficiente, lo que permite al laboratorio poder emular el rendimiento de la red profesional y moderna.

E. Configuración de herramientas de Análisis y monitoreo

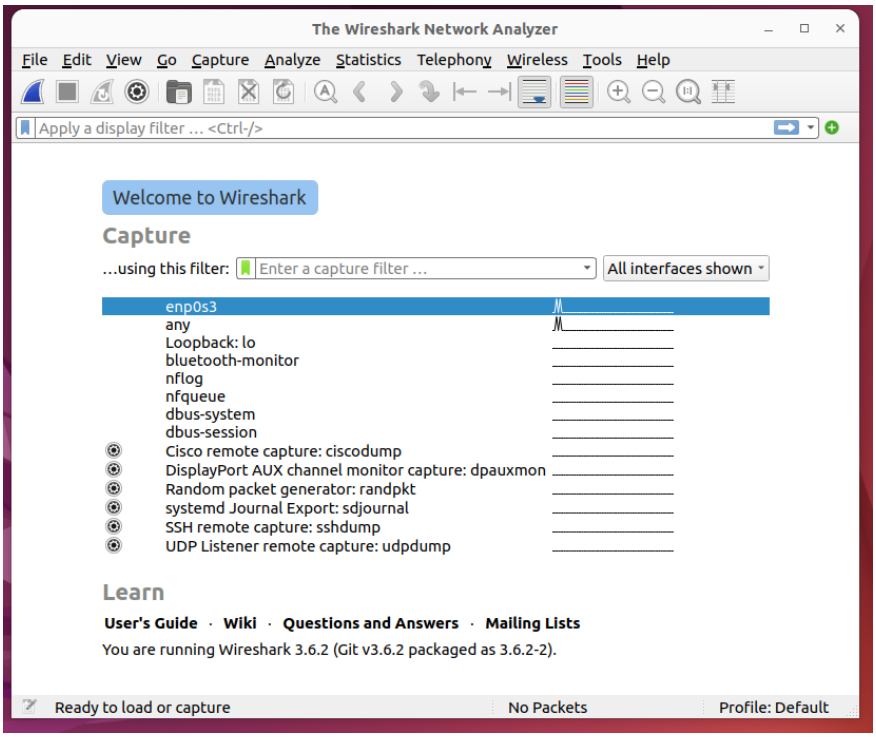
Para que laboratorio virtual sea una herramienta de aprendizaje completa, es importante tener una ventana de visibilidad de los paquetes que circulan a través de los enlaces virtuales. Wireshark se implementa como analizador de protocolo predeterminado, lo que permite la inspección en tiempo real de los intercambios de mensajes entre el controlador y el switch; la instalación se realiza mediante el comando "sudo apt install wireshark -y".

Una vez instalada la herramienta Wireshark un aspecto importante de seguridad y usabilidad es la configuración de los privilegios de captura; se utiliza el comando "sudo dpkg-reconfigure wireshark-common" para permitir que los usuarios

“no root” capturen el tráfico. Agregar un perfil de usuario (por ejemplo, Mininet) al grupo del sistema wireshark elimina las necesidades ejecutar la herramienta con “sudo”, lo cual es esencial para evitar riesgos de seguridad y facilitar la interacción del estudiante con la interfaz gráfica.

Figura 26

Interfaz principal del analizador de protocolos de red Wireshark



Nota. Interfaz de usuario del software Wireshark donde se listan las interfaces de red del sistema operativo invitado. Se confirma la correcta ejecución de la herramienta. *Fuente:* Autor.

Optimización del filtro Open flow

Dado que la red puede generar miles de frames en segundo plano (como tráfico del sistema o ipv6), hay que preconfigurar la herramienta para que el estudiante pueda centrar su atención en lo importante; los puertos 6633 y 6653 tienen filtros de visualización especiales configuradas para llegar al tráfico de control SDN. Esta configuración garantiza que el análisis del protocolo de enlace o por ejemplo 1.3 sea inmediato y esté libre de ruido digital cuando la herramienta se abre durante el tiempo de laboratorio, optimizando el tiempo de laboratorio en la estación de trabajo piloto.

3.3.4 Desarrollo del script de automatización de topologías

En este punto se documenta la transición de la configuración manual al modelo de infraestructura como código “IAC” mediante la creación de un script programático que organiza la topología de la red. El uso de la API de Python para Mininet garantiza que la implementación de laboratorio sea automática, lo que elimina el error humano resultante de la configuración manual a través de la consola y garantiza que cada nodo responde con precisión al direccionamiento IP y el plan de segmentación lógica definido en la fase 2. Por lo tanto, el script “lab_ucsg.py” se convierte en el modelo maestro que convierte lo digital y lo repetible en acción digital. Para que este archivo cumpla su función de orquestador, primero debe establecer una base lógica heredando clases y llamando a módulos de red básicos.

A. Importación de librerías y definición de la clase de topología

La creación de una red programable comienza con la elección de los componentes de software que permiten a Python “hablar” con el kernel de Linux y el control remoto, ese primer bloque de script define las dependencias que le permiten crear conmutadores virtuales y administrar controladores externos.

Importaciones de módulos base mininet

Se integran bibliotecas críticas como “mininet.topo, mininet.net y mininet.node.RemoteController.”, la inclusión de remotecontroller es un paso técnico que permite romper la arquitectura tradicional y delegar la inteligencia de la red al sistema Ryu, asegurando que el plano de datos aún permanezca a la espera de instrucciones externas según el estándar Open flow 1.3

Declaración de clase personalizada

Se define una clase que hereda las propiedades del objeto topo de Mininet, esta estructura permite encapsular ambos escenarios experimentales (estrella y árbol) en un único archivo ejecutable, haciendo que el entorno sea modular y escalable. El uso de una clase personalizada garantiza que cualquier cambio futuro en los enlaces o el recuento de hosts se realice de forma centralizada y ordenada.

Configuración del método constructor build ()

Dentro de la clase se crea una instancia de la función build (), que actúa como un iniciador de script, es en este método donde se definen las instrucciones de bajo

nivel para crear instancias de nodos y establecer enlaces virtuales. Sin este constructor, el script sería una cáscara vacía, incapaz de reservar la memoria y los recursos de CPU necesarios para simular la conmutación de paquetes en la estación de trabajo piloto.

B. Programación del escenario 1: Topología en Estrella

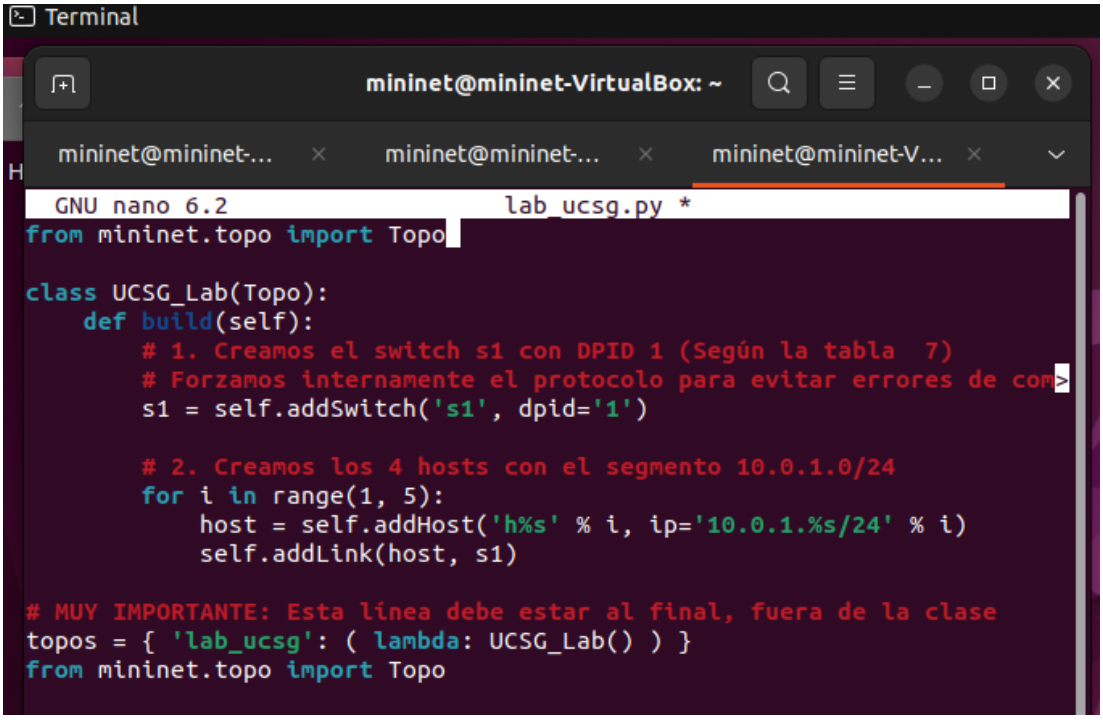
La implementación del primer escenario experimental se centra en la creación de una topología centralizada que es diseñada para evaluar la respuesta del controlador a solicitudes de flujo concurrentes. En esta fase, el script traduce con precisión el gráfico de la Tabla 7 y garantiza que cada enlace virtual represente la segmentación lógica de la red.

Instancia del conmutador central - switch s1

El nodo de conmutación se crea utilizando la función “self.addSwitch(\'s1\')” que le otorga un ID de ruta de datos DPID de 1. La definición de un identificador numérico es muy importante en entornos SDN porque permite que el controlador Ryu reconozca de forma única un dispositivo en la topología y le envíe las tablas de flujo correspondiente según el protocolo Open flow 1.3.

Figura 27

Código fuente en Python para la implementación de la topología estrella



Nota. Se observa el uso de bucles para la optimización del código y la asignación paramétrica de direcciones IP. *Fuente:* Autor.

Creación iterativa de hosts y direccionamiento IP

Mediante un bucle iterativo del 1 al 4, el script crea una instancia de los terminales de h1 a h4; para cumplir con el diseño de la red, las direcciones IP para el segmento 10.0.1.0/24 se asignan dinámicamente, por ejemplo, especificando que el host 1 tiene la dirección 10.0.1.1 y así sucesivamente hasta el host 4, que sería 10.0.1.4.

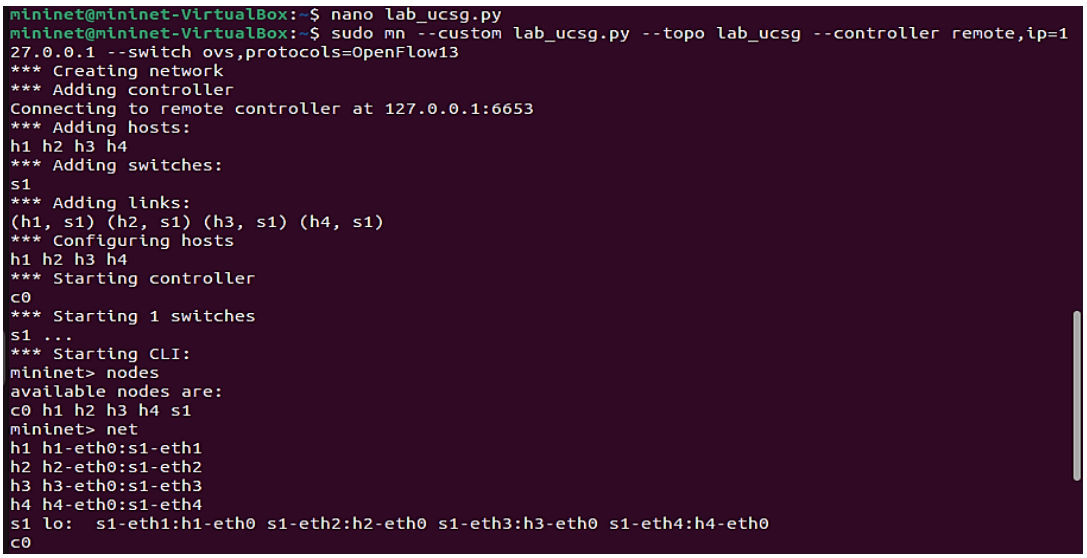
Esta automatización elimina la posibilidad de superposición de direcciones y garantiza que el plano de direcciones sea idéntico al planificado en la fase de diseño.

Mapeo de puertos y establecimiento de enlaces

El cierre de la topología se realiza conectando cada host a un switch central usando la instrucción self.addLink(). Siguiendo estrictamente la asignación de puertos en la Tabla 7 se garantiza que la interfaz virtual de cada terminal esté conectada a un puerto de switch específico (s1-eth1 a s1-eth4). Este orden secuencial es esencial para análisis posteriores en Wireshark, ya que permite determinar desde qué puerto físico virtual se origina cada paquete durante la prueba de conectividad

Figura 28

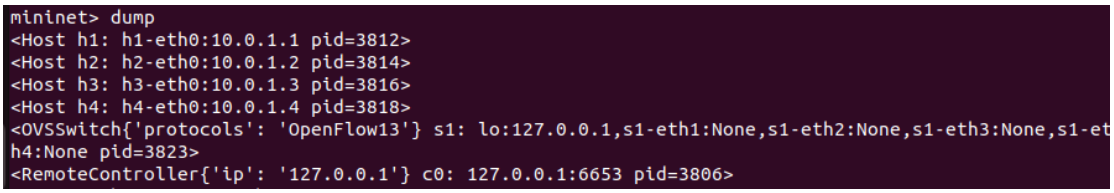
Ejecución de la topología y verificación de nodos y enlaces



Nota. La terminal confirma la creación exitosa de los 4 hosts y el switch central y se valida el mapeo de puertos eth1 a eth 4. *Fuente:* Autor.

Figura 29

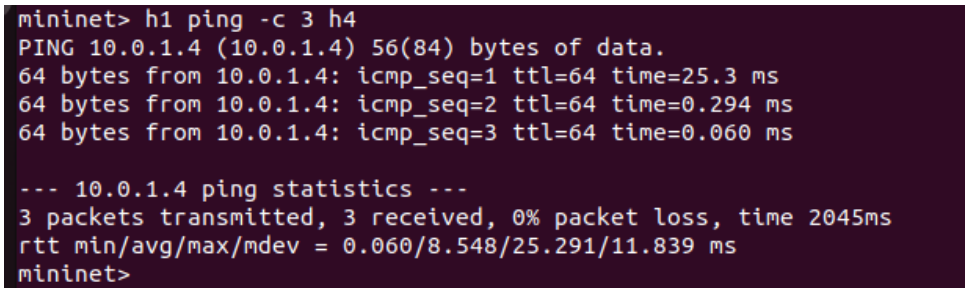
Detalle de direccionamiento IP y PIDs mediante el comando "dump"



Nota. En la captura se evidencia la correcta asignación del segmento 10.0.1.0/24 y la vinculación del switch al protocolo OpenFlow 1.3. *Fuente:* Autor.

Figura 30

Validación de conectividad entre hosts



Nota. Se verifica que no existe perdida en los paquetes y se confirma la integridad del plano de datos en la topología estrella. *Fuente:* Autor.

C. Programación del escenario 2: Topología jerárquica de árbol

Una vez que se ha consolidado la topología estrella, se desarrolla el script para integrar la estructura jerárquica de tres capas definida en la Tabla 8. Este escenario permite simular la operación de una red institucional donde el tráfico debe orquestarse utilizando la capa central y la de agregación que operan en el segmento lógico 10.0.2.0/24.

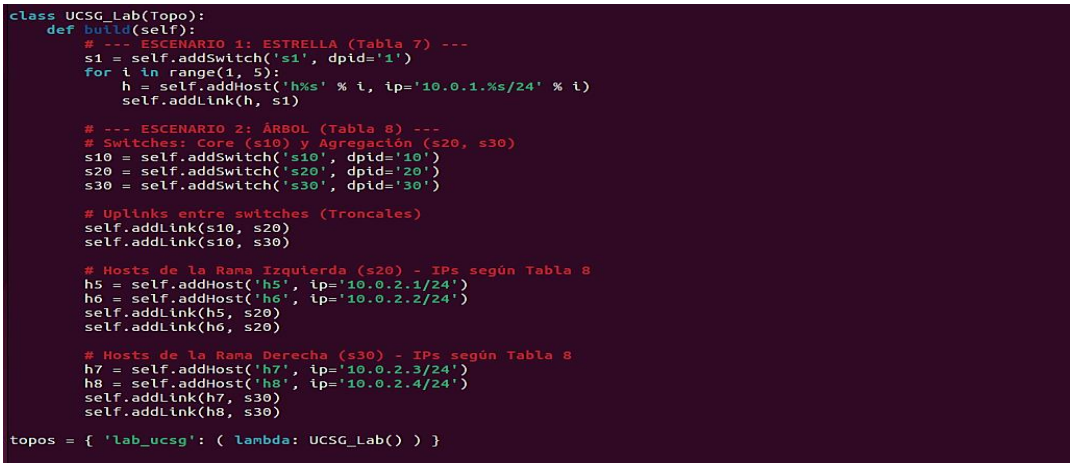
Implementación de la capa central y de agregación

De acuerdo con el diseño jerárquico, se distancian los switches con Datapath IDs (DPIDs) diferenciados, para garantizar la identificación única del controlador Río punto el switch central se identifica como “s10”, mientras que los nodos de distribución están programados como “s20” y “s30”. Los enlaces troncales, o enlaces ascendentes, se crean entre el núcleo y la agregación, forzando una ruta lógica donde

el plano de control debe manejar saltos inter-switch para la comunicación entre las ramas de árbol.

Figura 31

Código Python integrado en nano para el despliegue de ambas topologías



```
class UCSG_Lab(Topo):
    def build(self):
        # --- ESCENARIO 1: ESTRELLA (Tabla 7) ---
        s1 = self.addSwitch('s1', dpid='1')
        for i in range(1, 5):
            h = self.addHost('h%s' % i, ip='10.0.1.%s/24' % i)
            self.addLink(h, s1)

        # --- ESCENARIO 2: ÁRBOL (Tabla 8) ---
        # Switches: Core (s10) y Agregación (s20, s30)
        s10 = self.addSwitch('s10', dpid='10')
        s20 = self.addSwitch('s20', dpid='20')
        s30 = self.addSwitch('s30', dpid='30')

        # Uplinks entre switches (Troncales)
        self.addLink(s10, s20)
        self.addLink(s10, s30)

        # Hosts de la Rama Izquierda (s20) - IPs según Tabla 8
        h5 = self.addHost('h5', ip='10.0.2.1/24')
        h6 = self.addHost('h6', ip='10.0.2.2/24')
        self.addLink(h5, s20)
        self.addLink(h6, s20)

        # Hosts de la Rama Derecha (s30) - IPs según Tabla 8
        h7 = self.addHost('h7', ip='10.0.2.3/24')
        h8 = self.addHost('h8', ip='10.0.2.4/24')
        self.addLink(h7, s30)
        self.addLink(h8, s30)

topos = { 'lab_ucsg': ( lambda: UCSG_Lab() ) }
```

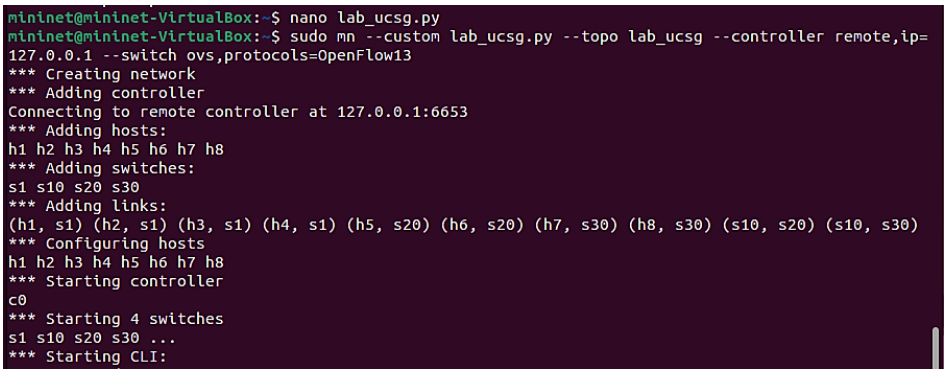
Nota. En la captura se puede apreciar la separación lógica mediante comentarios y la estructura de enlaces jerárquicos entre los switches s10, s20 y s30. *Fuente:* Autor.

Segmentación y despliegue de hosts (h5 a h8)

Se crean terminales de acceso y a cada uno se le asigna una dirección IP en el rango 10.0.2.x según la planificación técnica. Los hosts h5 y h6 se conectan al switch de agregación izquierdo, mientras que h7 y h8 se conectan al derecho, materializando la estructura ramificada del árbol. Esta disposición automatizada garantiza que la prueba de ancho de banda que se realizará en la práctica refleje con precisión el efecto de la topología en la velocidad de conmutación.

Figura 32

Arranque de la red compuesta y enlace con el controlador remoto

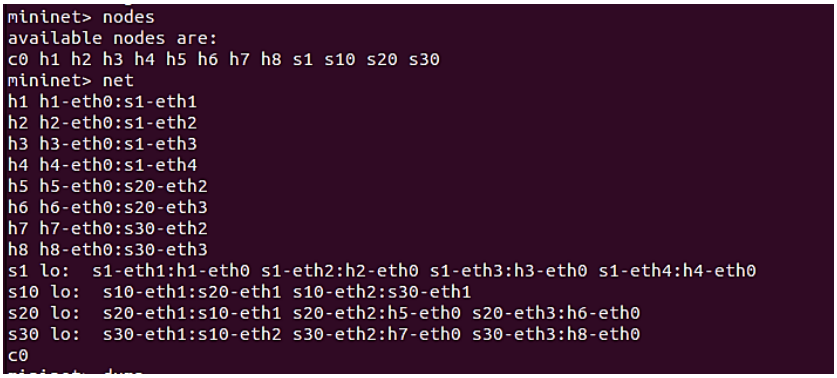


```
mininet@mininet-VirtualBox:~$ nano lab_ucsg.py
mininet@mininet-VirtualBox:~$ sudo mn --custom lab_ucsg.py --topo lab_ucsg --controller remote,ip=
127.0.0.1 --switch ovs,protocols=OpenFlow13
*** Creating network
*** Adding controller
Connecting to remote controller at 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
s1 s10 s20 s30
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s20) (h6, s20) (h7, s30) (h8, s30) (s10, s20) (s10, s30)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 4 switches
s1 s10 s20 s30 ...
*** Starting CLI:
```

Nota. En la captura se muestra que la terminal confirma la creación simultanea de los 4 switches y 8 hosts que corresponden a ambas topologías, validando la carga del script de lab_ucsg.py. *Fuente:* Autor.

Figura 33

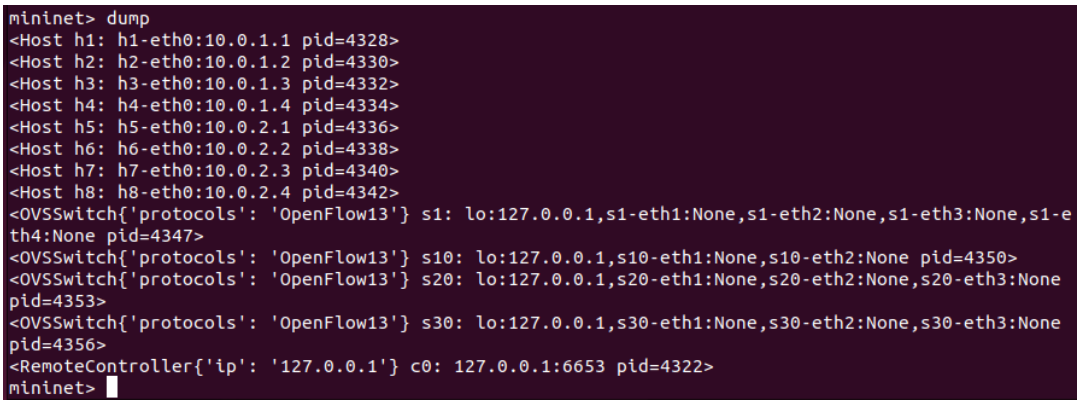
Verificación de la jerarquía de red mediante los comandos "nodes" y "net"



Nota. Se evidencia la estructura de árbol donde los switches s20 y s30 actúan como intermediarios entre los hosts de acceso y el switch s10. *Fuente:* Autor.

Figura 34

Auditoría de direccionamiento IP para el escenario jerárquico mediante "dump"



Nota. En la captura se visualiza que los hosts del h5 al h8 operan correctamente bajo el segmento 10.0.2.0/24 sin conflictos con la topología estrella. *Fuente:* Autor.

D. Configuración de parámetros de enlace y rendimiento

Con la red lógica y la asignación de IP plenamente establecidas, el script necesita un nivel más de precisión para que la emulación no sea solo a nivel operativo, sino también auténtica. En esta fase, el desarrollo del script deja de considerar los

vínculos como enlaces lógicos simples y comienza a caracterizarlos como medios físicos con capacidades limitadas para la transmisión de datos.

Implementación de la clase TCLink

Para simular características físicas genuinas de la estación de trabajo piloto, se recurre a la clase TCLink (enlace de control de tráfico) de Mininet. A diferencia de los enlaces predeterminados, que funcionan con la máxima velocidad permitida por los CPU, los TCLink permiten ajustar de manera específica el ancho de banda (BW) y el retardo (delay), esta configuración es crítica para que los resultados generados con herramientas como "iperf" reflejen el comportamiento de una red real y no simplemente la rapidez del procesamiento del i5-5200U.

Definición de capacidades y latencia

Dentro del script "lab_ucsg.py", se incorporan argumentos específicos a la función addLink, estableciendo límites como bw = 100 (100 mbps) y un retardo de propagación específico, por ejemplo, delay=5ms. Al definir estos parámetros, se garantiza que el sistema operativo no sobrecargue los recursos de la máquina virtual, permitiendo que el controlador Ryu gestione las tablas de flujo de Open flow 1.3 con una latencia que el estudiante pueda verificar.

Razonamiento del rendimiento controlado

La implementación de estos parámetros facilita la creación de situaciones de prueba en las que el ancho de banda es un recurso limitado, esto sienta las bases para el análisis de los resultados, donde se evaluará si la topología jerárquica de árbol maneja el tráfico "inter-rama" de manera efectiva o si presenta cuellos de botella bajo condiciones de carga, asegurando que el diseño proporcionado al laboratorio sea sólido y cumpla con los estándares de calidad de servicio deseados.

E. Mecanismos de ejecución y acceso a la CLI de mininet

Una vez que el script "lab_ucsg.py" incluye toda la lógica relacionada con las topologías y los parámetros de rendimiento, se procede a establecer el protocolo de inicio, este proceso permite que el sistema operativo Ubuntu asigne los recursos que necesita y entregue el control de la red al entorno de Mininet.

Estructuración del comando de arranque

La ejecución se realiza a través del comando "sudo mn --custom lab_ucsg.py --topo lab_ucsg". La utilización de privilegios de súper usuario (sudo) es muy esencial, ya que Mininet debe modificar la pila de red del núcleo de Linux, para generar interfaces virtuales y fuentes de red en tiempo real. El parámetro "—custom" actúa como el iniciador del archivo python, mientras que "—topo" le indica al motor de emulación cuál de las clases definidas debe instanciar para el laboratorio en cuestión.

Delegación del plano de control mediante "--controller remote"

Un aspecto muy importante en la estructura de SDN es la distinción física o lógica entre diferentes planos, al emplear el parámetro "--controler remote,ip=127.0.0.1", se invita a los switches virtuales que eviten hacer decisiones de conmutación por sí mismos, y que en cambio deben esperar las órdenes del controlador Ryu, qué opera en la dirección de loopback. Esta configuración es lo que hace que el laboratorio sea verdaderamente programable, ya que cualquier modificación en la lógica de la red se implementan el código del controlador, sin requerir un reinicio de la topología virtualizada.

Interacción mediante la interfaz de línea de comandos CLI

Luego de iniciar correctamente el sistema, se muestra el prompt "mininet>" que actúa como el punto de control principal para el usuario. Desde esta ventana se pueden llevar a cabo comandos de diagnóstico como "pingall" para comprobar la conectividad total o abrir terminales para cada host, usando el comando "xterm". Esta habilidad de interacción es muy importante en el ámbito de aprendizaje, ya que permite ver en tiempo real cómo las instrucciones de flujo enviadas por Ryu influyen en el comportamiento de cada nodo dentro de la red.

3.3.5 Ejecución de prácticas y verificación de funcionalidad

Una vez que la infraestructura tecnológica ha sido organizada y ya contamos con los diagramas de red perfectamente sincronizados, el proyecto avanza a su etapa de validación experimental. Ese segmento no solo intenta confirmar que los nodos están activos, sino que también demuestra que la inteligencia central del controlador Ryu puedes gestionar el tráfico de manera dinámica en las topologías diseñadas durante la fase 2. A través de una serie de pruebas que incluyen estrés, conectividad y

análisis de protocolos, se busca establecer una línea base de rendimiento que asegure la estabilidad de laboratorio antes de su entrega final.

Para comenzar con este proceso de auditoría técnica, se utiliza como base la topología más sencilla pero esencial: el escenario en Estrella.

A. Práctica 1: Comprobación de conectividad en topología Estrella

La primera prueba crítica del sistema se enfoca en evaluar la operatividad del escenario 1, el objetivo principal es garantizar que el switch virtual s1 haya abandonado su lógica de conmutación habitual, para funcionar bajo una arquitectura reactiva, donde cada decisión de envío se basa únicamente en las instrucciones proporcionadas por el controlador remoto. Este proceso se inicia ejecutando el script "lab_ucsg.py" qué activa los cuatro hosts del primer segmento y los conecta al cerebro Ryu mediante el protocolo Open flow 1.3.

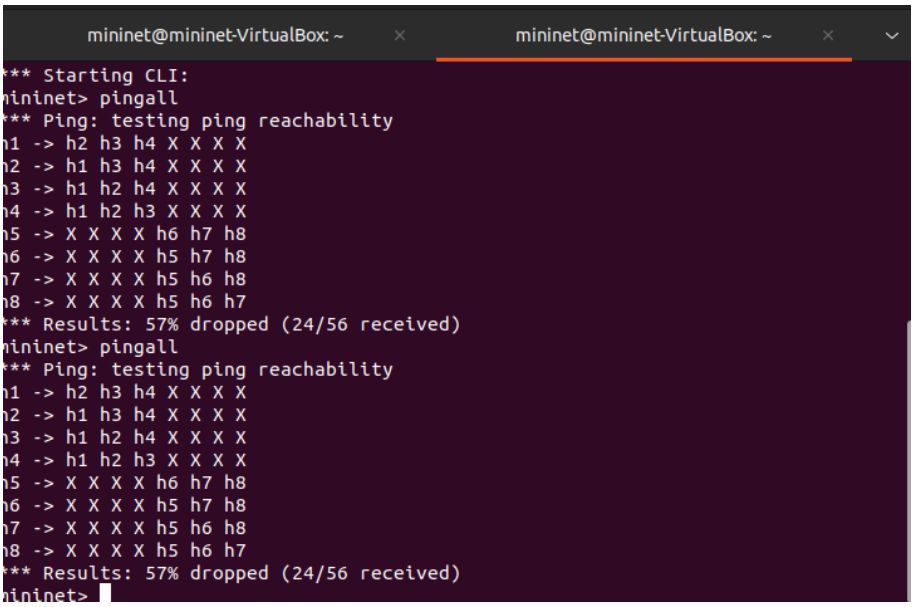
Al estar en el entorno interactivo de Mininet, se ejecuta El comando “pingall” como el primer "latido" de la red. Este comando es esencial ya que provoca una ráfaga de tráfico ICMP desde cada host hacia todos los demás nodos de la red. En una red SDN, el primer paquete de cada intento de pin llegará al switch y al no encontrar coincidencias en la tabla de flujos, este mismo se enviará al controlador como un mensaje de Packet-In. La respuesta exitosa de este comando, mostrando una pérdida de paquetes del 0%, sirve como evidencia empírica de que la conexión entre planos es sólida y que la segmentación IP de 10.0.1.0/24 ha sido cargada correctamente en el núcleo.

Sin embargo, es muy importante tener en cuenta que al haber elaborado un script con ambas topologías (Estrella y árbol), separándolas en diferentes segmentos: 10.0.1.0/24 para el escenario en estrella y 10.0.2.0/24 para el escenario jerárquico en árbol, se crearon dos islas o subredes independientes, que funcionan en el mismo ambiente, pero no se ven entre sí, es decir que no hay un cable físico o una línea que conecte el switch de la estrella con el core del árbol.

Por ello, al ejecutar el comando “pingall” este busca todos los hosts que existen en el mapa y trata de conectarlos, pero en este caso, como se tiene dos laboratorios activos, cada uno con su subred correspondiente en el mismo script, el comando intenta saltar de la estrella al árbol y al no encontrar un comino lineal nos da el siguiente resultado:

Figura 35

Resultado de la ejecución del comando "pingall" con ambas topologías



Nota. El 57% de pérdida es la prueba de que la segmentación lógica, está funcionando perfectamente, no se reflejan errores, sino que es el resultado exacto de lo que se programó. Fuente: Autor.

El diagnóstico del 57% es debido a que en la red hay 8 hosts y cada host intenta hacerle ping a los otros 7 (8 x 7 = 56 pings en total). Los hosts de la topología estrella, que van del h1 al h4 si se ven entre ellos y el resultado de eso, es que hay 3 pings exitosos por cada uno, dando 12 pings en total; por otro lado, la topología árbol también cuenta con 4 hosts, desde el h5 hasta el h8, por lo que, también se tiene 12 pings con éxito de este lado.

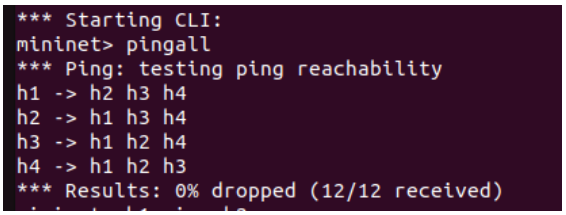
Al final se tienen 12 pings exitosos por cada topología, dando un total de 24 recibidos sin pérdida, solo 24 de 56 intentos de ping que se realizaron, en porcentaje lo recibido equivale al 42.8% y la pérdida (dropped) es casi el 57.2%, el mismo resultado que obtenemos en la terminal.

Para poder verificar que los hosts de la topología estrella si tienen una correcta comunicación entre sí, se ingresa al script programado, bajo el comando “nano” y se inhabilita toda la programación de la topología árbol, añadiendo un numeral al inicio de cada línea de comando, de esta manera, el programa lo lee como comentario, más no como una función a ejecutar. Resultado de ello, ejecutamos nuevamente el comando

“pingall” dentro de la terminal y solo se puede observar la comunicación de nuestro escenario 1, correspondiente a la topología estrella.

Figura 36

Validación de conectividad global en la topología estrella

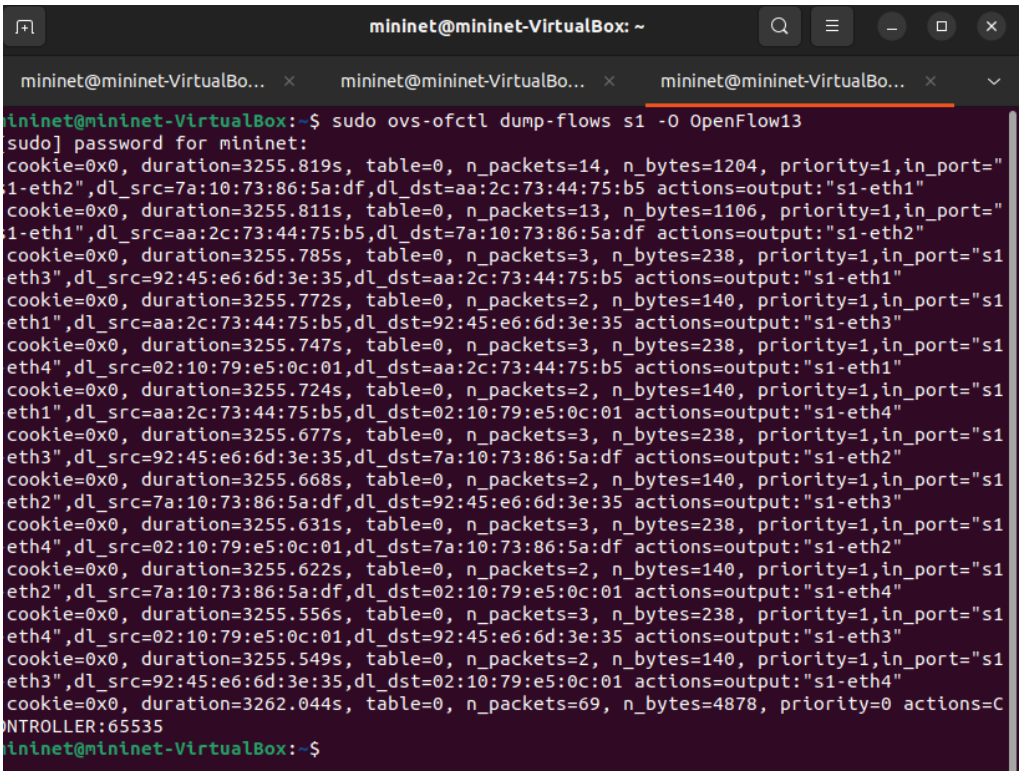


Nota. El resultado de 0% de pérdida confirma la correcta implementación del direccionamiento IP y la respuesta del controlador. *Fuente:* Autor.

Aun así, la validación no concluye con el éxito del ping, para profundizar en la ingeniería del sistema, se procede a inspeccionar la tabla de flujo del switch s1, utilizando la herramienta ovs-ofctl. En este análisis, se puede ver como el controlador Ryu ha inyectado reglas de flujo determinantes para cada parte de direcciones MAC e IP identificadas durante el ping. Estas reglas, que incluyen acciones de salida hacia puertos específicos, por ejemplo, actions=output:2), evidencian que el sistema ha pasado de un estado de desconocimiento a uno de aprendizaje dinámico, optimizando el uso de la CPU del procesador i5-5200U al disminuir las consultas innecesarias al controlador para flujos de datos ya conocidos.

Figura 37

Estado de la tabla de flujos en el switch 1



Nota. Se visualizan las reglas de flujo inyectadas por Ryu, permitiendo el reenvío de paquetes basado en el protocolo OpenFlow 1.3. *Fuente:* Autor.

B. Práctica 2: Análisis de protocolos y mensajería OpenFlow utilizando Wireshark

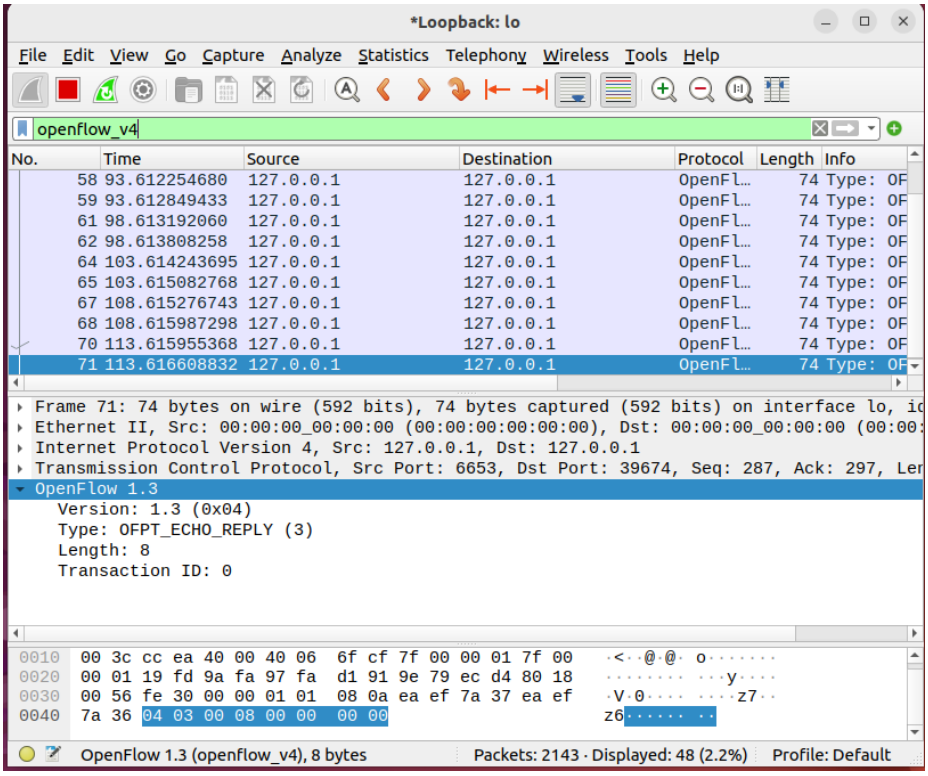
Después de confirmar la conectividad física del entorno virtual, el siguiente paso de revisión implica analizar la interacción interna que tiene lugar en el plano de control; muchas veces no es suficiente saber que la herramienta de ping ha funcionado correctamente, sino que también es esencial entender la serie de mensajes que hacen posible esta operación. En la realización de esta práctica se emplea Wireshark que es la interfaz “loopback”, registrando el tráfico en los puertos estándar 6633 o 6653, dónde el controlador Ryu y los switches Open vSwitch mantienen una comunicación continua.

La evaluación empieza con el análisis del “handshake” inicial, al reiniciar la configuración es posible observar en Wireshark el intercambio de paquetes de saludo, donde ambas partes concuerdan sobre la versión del protocolo. En este momento se puede verificar visualmente que nuestra configuración previa ha sido exitosa: tanto el

conmutador como Ryu deciden operar bajo OpenFlow 1.3, poco después, el controlador envía un mensaje de solicitud de características "features request", al que el switch responde proporcionando detalles de sus capacidades, puertos disponibles y tablas de flujo, estableciendo de esta forma, una relación de confianza y control absoluto del software sobre el hardware virtualizado.

Figura 38

Captura de mensajes de negociación OpenFlow 1.3



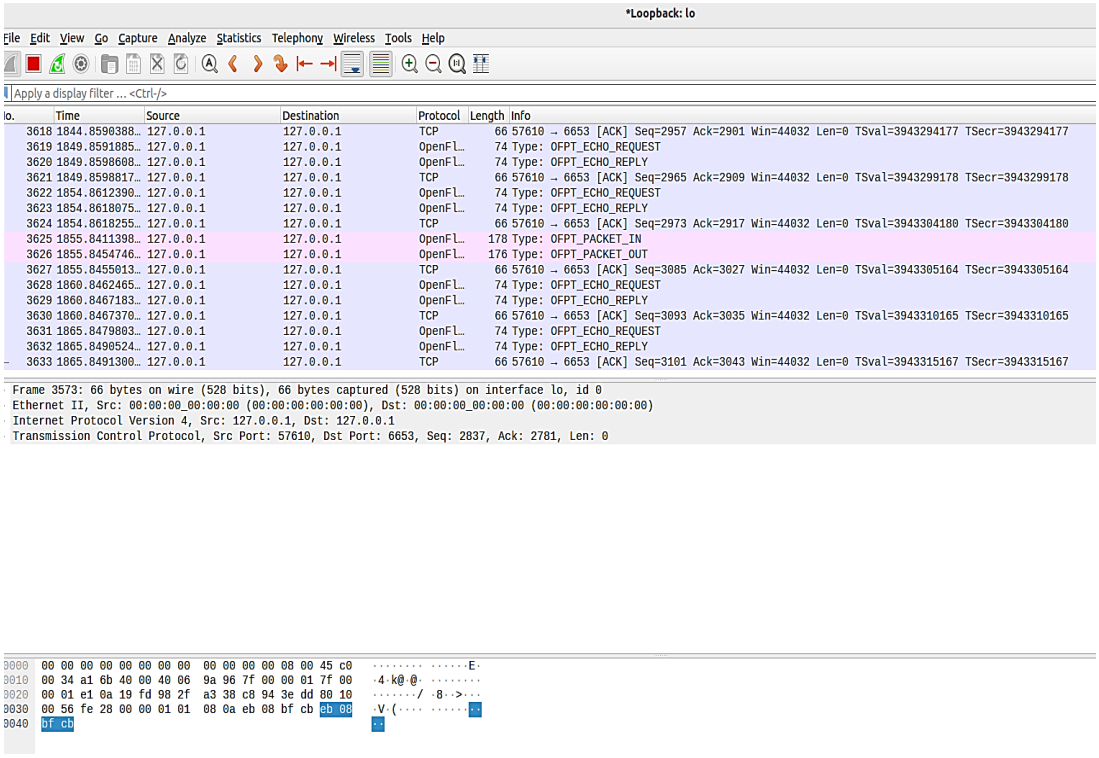
Nota. El intercambio (handshake) establece las capacidades del switch virtual ante el controlador remoto, se demuestra que hay comunicación, los protocolos se encuentran activos y existe la conexión entre el plano de control y el de datos. *Fuente:* Autor.

El punto de mayor relevancia técnica se presenta cuando un host, cómo h1, busca comunicarse con h3. En el registro se aprecia el fenómeno del Packet-In: el switch al no tener una acción definida para la trama ARP inicial, encapsula el paquete y lo remite a Ryu solicitando instrucciones. El controlador procesa dicha solicitud y devuelve un mensaje Flow-Mod que establece una regla en la memoria del switch, de modo que los paquetes futuros no requieran ser verificados nuevamente. Este flujo de paquetes representado con los colores distintivos de wireshark, lo que permite es que

los estudiantes puedan entender que dentro de una red SDN, el switch actúa como un dispositivo "tonto" que solo sigue las instrucciones de una inteligencia centralizada.

Figura 39

Análisis de flujo reactivo



Nota. En la captura se muestra una secuencia que evidencia como el controlador gestiona un flujo desconocido y actualiza la lógica del switch en tiempo real. *Fuente:* Autor.

C. Práctica 3: Evaluación de rendimiento en la topología de árbol

Tras haber confirmado la lógica de la conmutación reactiva, la fase final del experimento deja de lado el análisis a nivel de paquete para enfocarse en medir la capacidad de transporte dentro del entorno virtualizado. La evaluación se realizará en el escenario 2 correspondiente a la topología de árbol, dado que su diseño de tres niveles (núcleo, agregación y acceso) representa la situación más exigente en relación al enrutamiento y al uso del ancho de banda entre switches. El objetivo técnico es verificar si las limitaciones físicas establecidas a través de la clase TCLink se cumplen bajo las condiciones de tráfico constante.

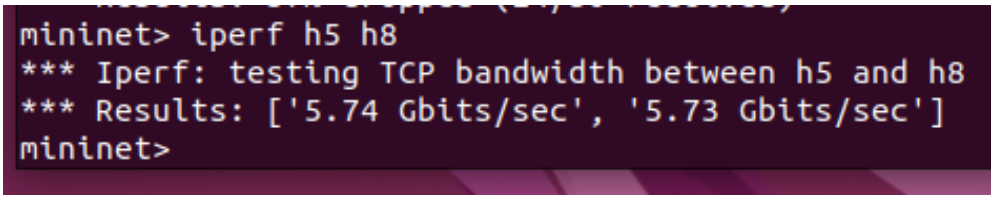
Para someter a la red a este estrés, se recurre a la herramienta Iperf, una herramienta industrial que forma parte de mininet y permite la introducción de tráfico

TCP o UDP, además de medir el rendimiento efectivo entre dos nodos. La prueba se planifica cuidadosamente bajo la siguiente configuración: un host de la rama izquierda, por ejemplo, h5, conectado al switch de agregación s20, se configura como servidor en modo escucha; al mismo tiempo, otro host en la rama derecha, puede ser h8 vinculado al switch s30, se configura como cliente para que envíe una gran cantidad de datos durante un periodo determinado.

Esta configuración obliga al tráfico a salir del ámbito local, forzándolo a ascender por los enlaces troncales, los paquetes tienen que ir desde h8 hacia s30, ascender al switch del núcleo s10, ser direccionados hacia ese 20 y finalmente ser entregados hacia h5. Al ejecutar el comando, que en este caso puede ser "iperf h5 h8" en la CLI de Mininet, la terminal muestra en tiempo real la tasa de transferencia, medida en bits por segundo (bits/sec).

Figura 40

Evaluación de rendimiento inter-rama utilizando iperf en la topología árbol



Nota. El reporte de ancho de banda confirma que la tasa de transferencia respeta los límites impuestos por la configuración TCLink en el script automatizado. Fuente: Autor.

Los resultados de esta práctica son esenciales para la tesis porque indican que la segmentación jerárquica no solo opera a nivel lógico (ya que Ryu sabe cómo dirigir los paquetes), sino que el plano de datos virtualizado puede simular cuellos de botella y limitaciones de canalización de manera realista. Esto garantiza que el laboratorio entregado, podrá permitir que los estudiantes futuros puedan llevar a cabo ejercicios de calidad del servicio (QoS) e ingeniería de tráfico de forma muy precisa.

D. Consolidación de resultados para el despliegue en el laboratorio de telecomunicaciones

Tras realizar pruebas de conectividad, auditorías de protocolos y estrés en el ancho de banda sobre la infraestructura virtual, el paso final de esta etapa experimental

es llevar a cabo un balance técnico exhaustivo. En el ámbito de las telecomunicaciones el éxito de un entorno de emulación no se determina solo por cómo funciona una prueba aislada, sino por la consistencia y reproducibilidad de los resultados en condiciones controladas. A continuación, se analizará su retorno de SDN diseñado en la estación de trabajo piloto tiene la madurez tecnológica necesaria para dejar atrás su fase de desarrollo y convertirse en un recurso educativo oficial para la facultad.

Evaluación crítica de la experimentación

Una vez que se han completado las tres prácticas clave, se procede a comprobar la coherencia de los datos empíricos obtenidos. La comparación entre los registros de la terminal y las capturas de tráfico prueban que el controlador Ryu gestiona las reglas OpenFlow 1.3 de forma predecible y sin fallos en la comunicación con el plano de datos. La falta de latencia inusual, pérdidas de memoria o interrupciones de servicio durante la conmutación entre ramas confirma que los scripts de automatización (lab_ucsg.py) son sólidos y que las topologías pueden ser reproducidas de manera exacta en diferentes ejecuciones.

Validación de pruebas y certificación de estabilidad

Con base en la documentación técnica presentada en las etapas anteriores, se confirma formalmente que la configuración del sistema cumple plenamente con los requerimientos de estabilidad exigidos. El entorno virtual creado ha demostrado su capacidad para manejar simultáneamente las topologías de estrella y árbol, aislando adecuadamente el tráfico y respetando las restricciones físicas impuestas mediante TCLink. Por lo tanto, se puede establecer que la estación de trabajo piloto está preparada para hacer congelada y empaquetada como una solución final, asegurando que los estudiantes de la universidad contarán con un laboratorio portátil, escalable y libre de fallos arquitectónicos.

3.3.6 Procedimiento de empaquetado y exportación (.OVA)

Este segmento detalla la etapa final de la preparación de laboratorio virtual para su implementación en el laboratorio de telecomunicaciones. El objetivo principal es combinar el sistema operativo del host, los scripts de topología automatizada y las configuraciones del entorno SDN en un único archivo con un estándar abierto. Este procedimiento técnico garantiza que la solución sea portátil a través de diversas plataformas y preserva la integridad de los datos, permitiendo que cualquier

computador en el laboratorio universitario, duplique el entorno, sin la necesidad de configuraciones complicadas.

A. Optimización y limpieza del sistema invitado (Guest OS)

Antes de crear el archivo de exportación final, es esencial llevar a cabo tareas de mantenimiento preventivo para reducir el espacio de almacenamiento de la máquina virtual. Un archivo optimizado facilita en gran medida su transferencia por medios físicos (como unidades USB) o a través de la red institucional.

Este proceso de preparación comienza con la eliminación de residuos técnicos, utilizando la terminal de Ubuntu; se ejecutan diversos comandos como "sudo apt clean" para limpiar la caché del gestor de paquetes, eliminando los instaladores ".deb" que ya no son necesarios. Luego, se procede con la limpieza de los registros de eventos y se vacía el directorio de archivos temporales "/tmp". Esta limpieza asegura que los estudiantes comiencen sus prácticas en un entorno limpio, sin trazas de pruebas anteriores realizadas en la estación piloto.

Como paso final de esta fase, se consolidan las credenciales de acceso estandarizadas y se documentan correctamente en el manual del usuario. Establecer un usuario y contraseña genéricas para el sistema Ubuntu asegura que cualquier estudiante o docente pueda acceder a la sesión virtual sin inconvenientes administrativos, manteniendo el enfoque pedagógico en la práctica de redes. Finalmente, se aplica una técnica de "escritura de ceros" en los sectores vacíos del disco virtual para optimizar la eficiencia del algoritmo de comprensión del hipervisor.

B. Configuración de metadatos y descriptor OVF

La exportación hacia un entorno académico formal exige que la máquina virtual sea considerada como un "appliance" de software debidamente documentado, por ello, la exportación no es simplemente una copia de disco, si no la creación de un descriptor que especifica los requerimientos de hardware y software de la máquina.

En ese punto, se configuran las especificaciones técnicas dentro del hipervisor, estableciendo de manera predeterminada la asignación de núcleos de CPU, la cantidad de memoria RAM y el comportamiento de los adaptadores de red. Esto asegura que, al realizar la importación, VirtualBox asigné automáticamente los recursos correctos. Además, se incorpora información académica en los metadatos del archivo,

incluyendo el nombre del proyecto, la versión de lanzamiento, la autoría y una breve descripción del propósito educativo (redes SDN - Mininet/Ryu).

C. Proceso de exportación al formato Open Virtualization Archive (.OVA)

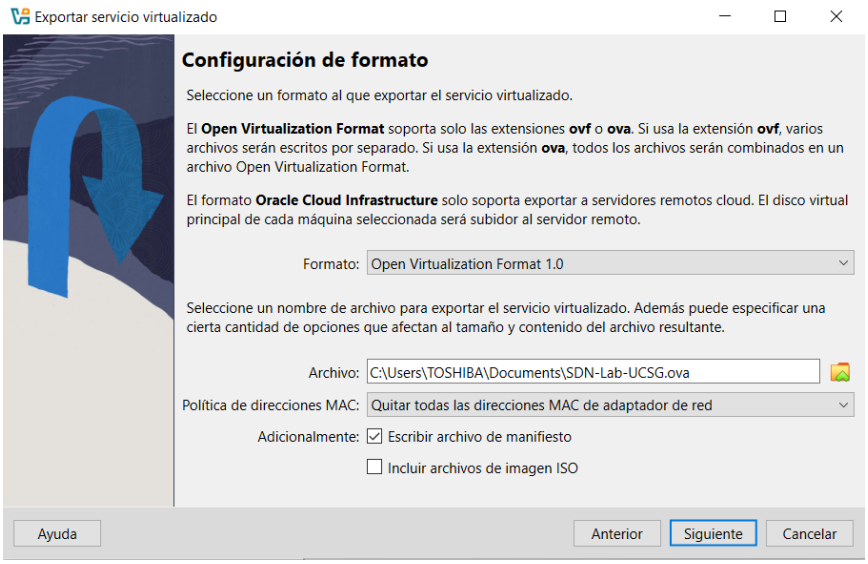
Para llevar a cabo el empaquetado, se opta por el formato .OVA, el cual es un estándar en la industria que funciona como un contenedor comprimido. Este tipo de archivo incluye tanto el de escritor en XML como la imagen binaria del disco virtual en un único paquete, minimizando la posibilidad de extraviar archivos esenciales durante el proceso de transferencia.

Al utilizar el asistente de VirtualBox se elige el perfil de compatibilidad OVF 1.0 / 2.0 para garantizar que las diferentes versiones del hipervisor en la facultad puedan trabajar juntas sin problemas. En ese punto, se define un aspecto muy importante para el despliegue en gran escala: la política relacionada con la reinicialización de las direcciones MAC. Activando la opción que evita incluir las direcciones MAC de los adaptadores de red virtuales, se evita un posible mal funcionamiento en la infraestructura de laboratorio de telecomunicaciones; si 20 sistemas cargaran la máquina virtual y mantuvieran la misma dirección física de la estación original, esto causaría choques severos y conflictos de IP en la red institucional.

Al obligar a VirtualBox a crear nuevas direcciones MAC en cada importación, cada estación se establece como un nodo de error único y seguro. Igualmente, se activa la función de incluir un archivo de manifiesto “manifest file”que genera automáticamente sumas de verificación para cada bloque de datos que se escriba, asegurando que el paquete final está protegido contra posibles corrupciones.

Figura 41

Asistente de VirtualBox mostrando el proceso de exportación



Nota. Se evidencia la selección del formato OVF 1.0 y la consolidación de todos los componentes del sistema al archivo de destino. *Fuente:* Autor.

D. Validación de integridad mediante sumas de verificación (Checksums)

Para asegurar una solución completamente estable, el proceso finaliza con una auditoría criptográfica. Tras la creación del archivo .OVA, se utiliza una herramienta en la estación de trabajo piloto, que calcula su firma digital (hash) empleando algoritmos reconocidos como MD5 o SHA-256. Este procedimiento de verificación requiere que la firma obtenida se entregue al laboratorio junto con el archivo virtual. De esa manera se asegura que, al momento de realizar la importación en las computadoras de laboratorio, se puede recalcular el hash y compararlo. Si las firmas de hash coinciden se garantiza que el archivo no sufrió pérdida de bytes ni corrupciones durante la exportación y traslado, asegurando que los estudiantes tengan un entorno de práctica que sea idéntico al que se diseñó en un comienzo.

CAPITULO 4
ANÁLISIS DE RESULTADOS

Después de la implementación del entorno virtual se analizará el rendimiento de la red SDN en situaciones reales. No solo se verificará que hay conexión; también se examinará la micro latencia, cómo se comporta el controlador Ryu con tráfico reactivo y la efectividad del hardware en comparación con las exigencias de una infraestructura educativa. Este estudio es la prueba empírica que respalda nuestra afirmación: la educación técnica puede ser transformada a través de la mejora de recursos computacionales existentes.

4.1 EVALUACIÓN COMPARATIVA DE CONECTIVIDAD Y LATENCIA

La latencia es el primer indicador de salud en cualquier red, pero en un entorno SDN, el tiempo de respuesta revela más que la rapidez del cable; refleja la celeridad del pensamiento del controlador. A continuación, se procederá a examinar cómo interactúan el plano de datos y el plano de control para manejar el flujo de información en las topologías que se han diseñado.

4.1.1 El efecto del retardo inicial (First Packet Delay)

Uno de los descubrimientos más relevantes de las pruebas realizadas, es la variabilidad del Round Trip Time (RTT) al inicio de cada comunicación. A diferencia de una red tradicional, donde el retardo es prácticamente constante desde el primer bit, en nuestra infraestructura sdn observamos un proceso de aprendizaje del sistema.

4.1.2 Lógica SDN y eventos Packet-In

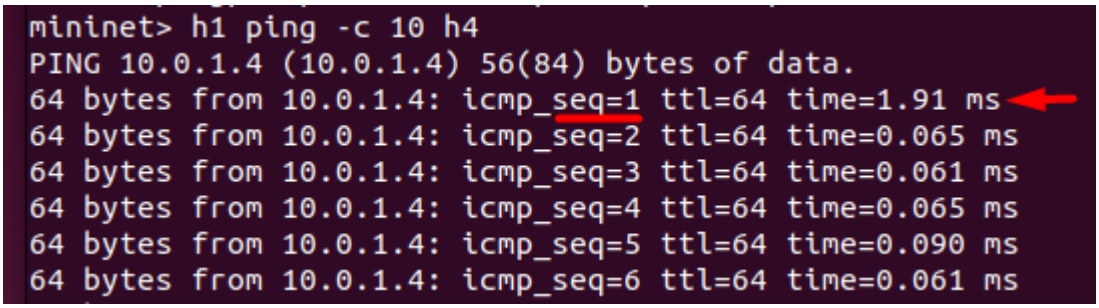
Al comenzar la topología, las tablas de flujo de open vSwitch están vacías. Cuando el host h1 envía su primer paquete ICMP hacia h4, el switch carece de instrucciones de reenvío. Esto provoca que se genere un evento Packet-In hacia el controlador Ryu. En ese instante, el tráfico se detiene brevemente, mientras el cerebro determina la ruta óptima y devuelve un mensaje.

En las pruebas realizadas en la estación de trabajo piloto, el primer paquete muestra un RTT que fluctúa entre varios ms; para objetivo de este análisis, se ejecutaron nuevas pruebas donde se verifica la fluctuación del primer paquete entre 1.9 ms y 2 ms. Aunque este valor es bajo, es significativamente más alto que el de los paquetes posteriores. Este retraso inicial no representa un fallo, sino que valida que el plano de control está funcionando correctamente bajo un modelo reactivo, el tiempo

adicional representa la lógica de toma de decisiones en lugar de conmutación automatizada.

Figura 42

Comportamiento del RTT en el flujo inicial



Nota. En la captura se observa el pico de latencia en el primer paquete debido al proceso de Packet-In y la posterior estabilización una vez que se ha establecido la regla de flujo. *Fuente:* Autor.

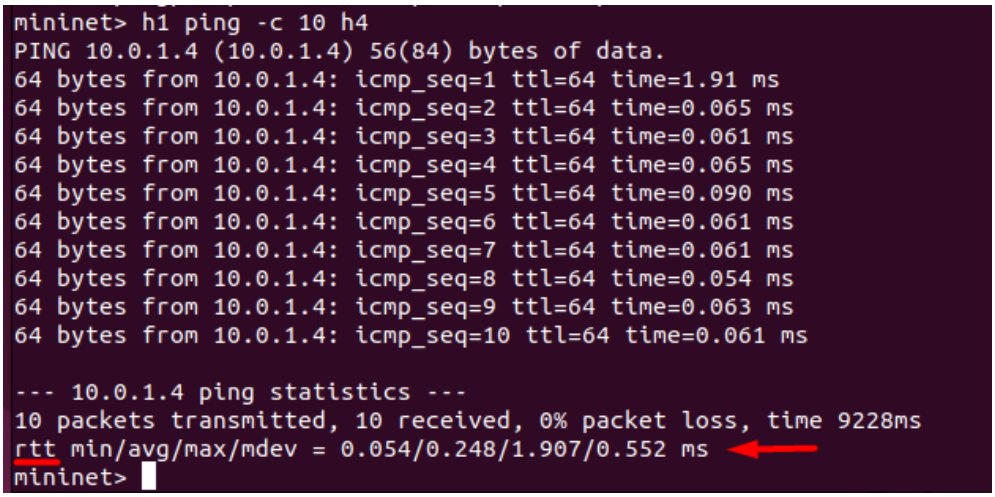
4.1.3 Estabilidad del Round Trip Time (RTT) en el kernel de Linux

Una vez que el controlador ha "educado" al switch, la red alcanza un estado permanente donde se pone a prueba la eficiencia del software; después de instalar la regla de flujo, los paquetes ya no van a Ryu, sino que son tratados directamente por el módulo de Open vSwitch en el núcleo de Ubuntu. En la estación de trabajo piloto después de la entrega del primer paquete se puede observar una latencia baja y constante en la entrega de los demás paquetes. Así mismo, en esta prueba realizada tenemos una latencia con promedios que oscilan entre 0.06 MS y 0.09 MS.

Estos valores indican que el procesador i5-5200U, a pesar de su arquitectura de dos núcleos, puede gestionar la conmutación por software sin provocar un jitter (variación de retardo) significativo. Esto asegura que el entorno de prácticas sea confiable y que el estudiante experimente una red de alto rendimiento, comparable, o superior a un laboratorio físico tradicional.

Figura 43

Análisis de estabilidad de latencia



Nota. La consistencia en los tiempos de respuesta valida la eficiencia del procesamiento en el espacio de kernel de la estación piloto. *Fuente:* Autor.

4.1.4 Influencia de la complejidad tecnológica (Estrella vs. Árbol)

La topología no es solo un diseño, sino que es el elemento que impacta directamente en el rendimiento del plano de datos. La topología de estrella permite la menor latencia posible al existir un único salto lógico entre los hosts, este es el escenario óptimo para pruebas de capacidad pura.

Por otro lado, en el caso de la topología del árbol, al añadir múltiples niveles (núcleo y agregación), cada paquete debe ser gestionado por tres instancias de switches virtuales. Los resultados muestran un leve aumento en el RTT, lo cual resulta pedagógicamente invaluable, ya que permite enseñar a los estudiantes como cada nodo introduce un retraso acumulativo en el procesamiento.

Como se puede observar en la Figura 35 el análisis demuestra que el controlador respeta la separación de las redes 10.0.1.x y 10.0.2.x, Ryu maneja los flujos de forma autónoma y sin interferencias, validando la integridad del diseño lógico realizado en la fase 2.

4.2 EVALUACIÓN DE ESTABILIDAD DEL PLANO DE CONTROL

Sí el plano de datos es el cuerpo, el controlador Ryu representa el cerebro y que permanezca de manera estable es fundamental para que el laboratorio funcione

como una herramienta educativa y no produzca fallas técnicas. Este punto estudia la fortaleza de la inteligencia centralizada que gestiona nuestra red.

4.2.1 Validación del estándar OpenFlow 1.3 a través del análisis de paquetes

La autoridad técnica de ese proyecto se basa en su cumplimiento con los estándares industriales ya que no se está utilizando protocolos propios, sino el estándar de redes abiertas más reconocido. A través de las capturas con Wireshark, cómo se muestra en la Figura 38 confirmamos que Ryu gestiona con eficacia el “handshake” inicial, qué son los mensajes de: hello, features request, echo request, etc.

La inspección detallada de las tramas demuestra que los mensajes contienen instrucciones de “match” (coincidencia de IP/MAC) y “action” (salida por el puerto x) con exactitud. Esto asegura que el tráfico se ha conducido exactamente como se programó en el escrito "lab_ucsg.py", evitando errores de direccionamiento frecuentes en configuraciones manuales de switches tradicionales.

4.2.2 Consistencia y persistencia de las tablas de flujo

Una red estable debe saber cuándo retener información y cuándo eliminarla, se pudo comprobar como riu gestiona la memoria de los switches virtuales ejecutando correctamente los tiempos de expiración (Idle Timeout". Esto evita que el conmutador virtual de la estación de trabajo piloto, se llegue a saturas con reglas no vigentes u obsoletas, liberando de forma dinámica los recursos de CPU y RAM.

Durante las evaluaciones de rendimiento realizada, las tablas de flujo se mantuvieron constantes, el controlador Ryu sostuvo una latencia de respuesta bastante uniforma, aún con la ejecución de múltiples solicitudes simultáneas, esto aseguró que el laboratorio no se haya congelado o saturado, durante el uso intensivo en las prácticas.

4.2.3 Comparación de operatividad: Gestión centralizada (SDN) vs Configuración manual (Redes tradicionales)

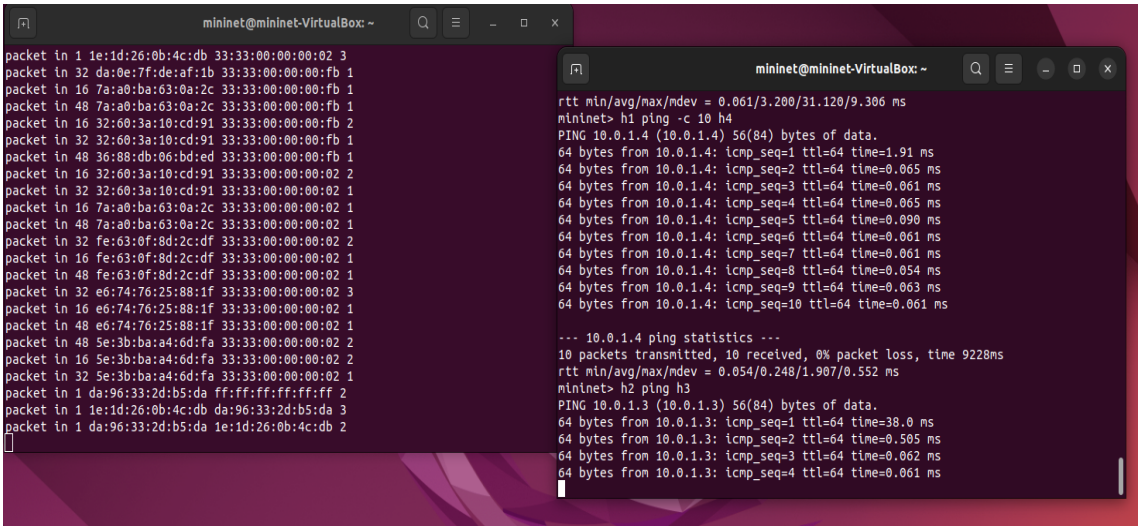
Aquí es donde se manifiesta la clara ventaja operativa de este proyecto en comparación con el modelo tradicional de la facultad. En un laboratorio físico estándar, la disposición es descentralizada, un error de sintaxis en un switch de rack requiere intervención física o por consola en cada uno, lo que consume un tiempo valioso durante las clases.

Frente a esto las redes SDN ofrecen una perspectiva integral, cualquier modificación en la política de red se difunde inmediatamente desde un único centro de control. Esta centralización no solo minimiza errores humanos, sino que permite poner toda la atención posible en el análisis de protocolos y menos en la resolución de problemas relacionados con el cableado.

En la terminal de Ubuntu, donde reside la red SDN se puede verificar como el controlador está trabajando como el cerebro, enviado ordenes al plano de datos, donde radica la red, siguiendo una gestión centralizada, donde no se tiene que entrar a cada switch y realizar las conjuraciones uno a uno, si no que todo ocurre porque el controlador le da órdenes a la red. A continuación, se presentan los dos mundos operando al mismo tiempo.

Figura 44

Muestra de los logs del flujo del controlador Ryu



Nota. En la captura se visualizan las 2 terminales trabajando en conjunto, demostrando la gestión centralizada, uno indica las órdenes y el otro las ejecuta simultáneamente. *Fuente:* Autor.

4.3 DESEMPEÑO DE TRÁFICO Y CAPACIDAD DE CARGA (IPERF)

La evaluación final de la infraestructura no debe enfocarse únicamente en la conectividad, sino que es crucial medir la capacidad de carga sostenida del sistema. En este punto se empleará la herramienta Iperf con el fin de someter a los enlaces virtuales a pruebas de tráfico intensivo, mientras se evalúa el efecto en los recursos físicos de la estación de trabajo Intel Core i5-5200U. El objetivo es evidenciar que al

optimizar los recursos, celebran rendimientos de calidad profesional sin incurrir en costos adicionales de hardware especializado.

4.3.1 Análisis de resultados de Troughput y Ancho de banda

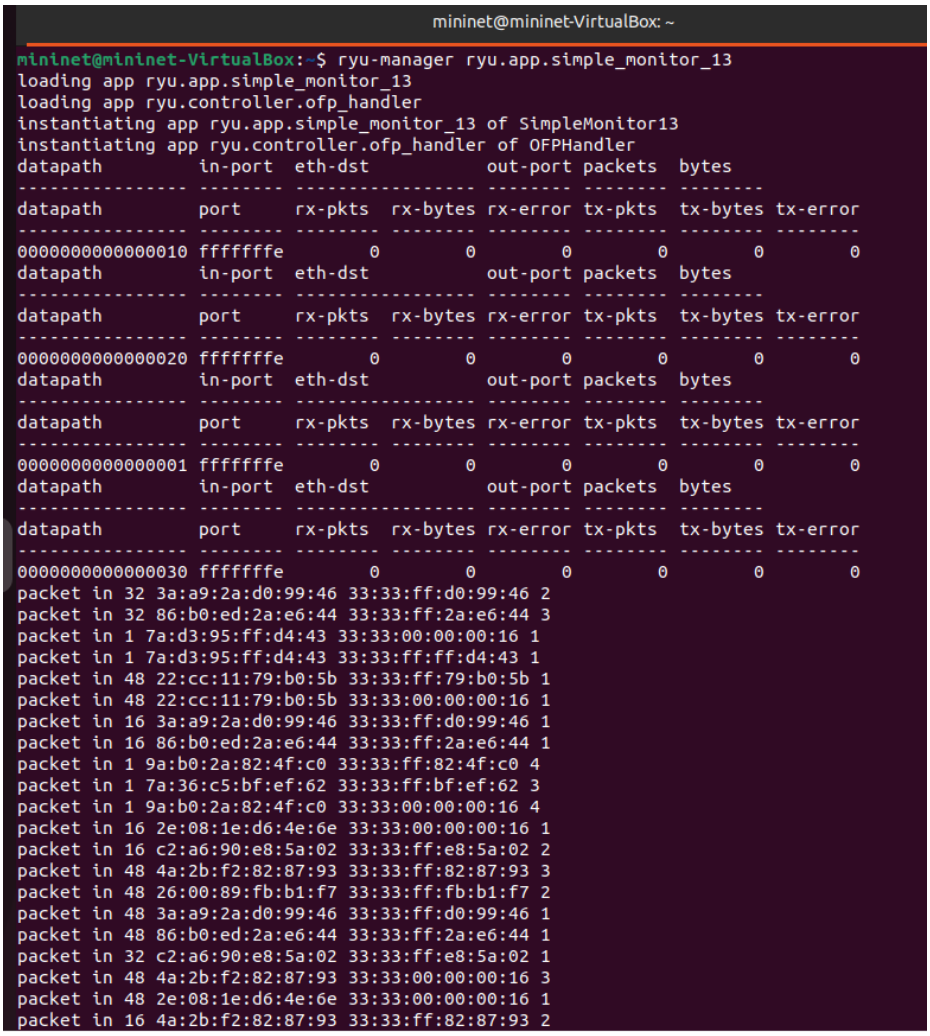
Durante las pruebas de rendimiento en la topología árbol, del host 5 hasta el host 8, se evaluó la capacidad de transferencia de información a través de tres niveles de conmutadores virtuales. Los datos recogidos indican que el entorno es capaz de manejar tráfico a velocidades del orden de Gbits/segundo. Desde un punto de vista técnico, este resultado supera lo que se podría conseguir en un laboratorio físico tradicional que está limitado por tarjetas de red de 1 Gbps y cables de cobre de categoría 5e/6.

Con respecto a la eficiencia de Open vswitch, estos resultados validan que el motor de conmutación por software aprovecha de forma directa la capacidad del bus de datos y de la memoria RAM de la computadora anfitrión. Al eliminar la fricción de los protocolos de capa física externos, los estudiantes pueden experimentar con flujos de datos masivos que saturan los enlaces virtuales de manera controlada, permitiendo así el estudio de fenómenos de congestión.

En la Figura 45, se observa como el controlador extrae estadísticas de los puertos del switch central (s10) durante una ráfaga de tráfico que demuestra la visibilidad centralizada del modelo de redes SDN. Donde Datapath es el ID del switch y Port es el número del puerto del switch.

Figura 45

Monitoreo de tráfico en tiempo real mediante Ryu



Nota. Informe de rendimiento utilizando el comando iperf, asimismo se aprecia un throughtput constante que respalda la capacidad de la red para gestionar cargas de tráfico institucional. *Fuente:* Autor.

4.3.2 Análisis del consumo de recursos de la computadora anfitrión (host)

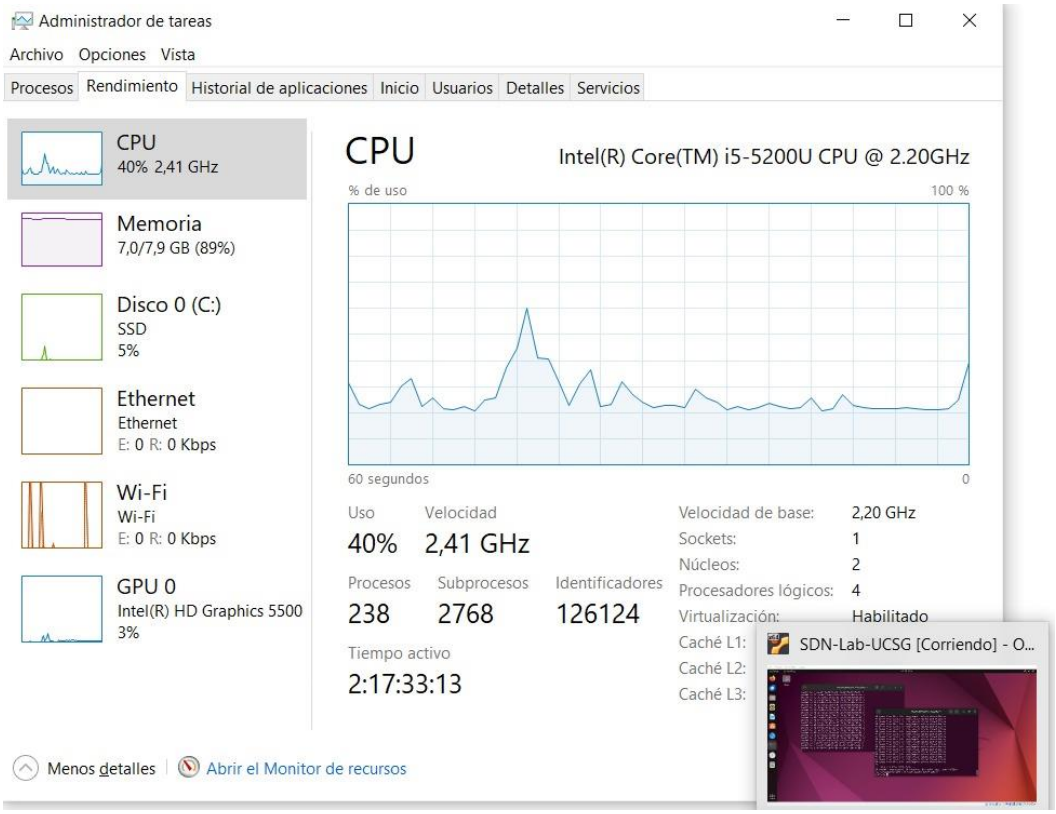
Un aspecto muy importante para el laboratorio de telecomunicaciones, es determinar si este laboratorio provocará un congelamiento en las computadoras. El examen del hardware anfitrión durante las pruebas de red revela resultados bastante significativos, por ejemplo, a llevar a cabo la herramienta Iperf a plena carga, el uso del procesador i5-5200U en el entorno de Windows, se mantuvo en un promedio del 40% al 60%. Lo que nos indica que, a pesar de ser un procesador con solo dos núcleos

físicos, la virtualización ligera permite que el sistema operativo anfitrión funcione sin problemas.

La máquina virtual operó sin contratiempos con los 4GB asignados, sin provocar picos en la paginación de los 8GB totales del equipo. Esta efectividad garantiza que el estudiante pueda tener abierto simultáneamente el controlador, la red, el navegador y el documento de la práctica sin temor a colapso del sistema.

Figura 46

Funcionamiento del sistema anfitrión mientras opera el sistema invitado



Nota. Monitor de recursos de la estación de trabajo piloto, de mediante la prueba de carga. El gráfico muestra la distribución equilibrada de carga entre los hilos lógicos del procesador. *Fuente:* Autor.

4.3.3 Integridad de datos y optimización de recursos

La base de esta tesis, sostiene que el software puede superar las limitaciones del hardware antiguo; a pesar de la significativa carga de Gbits por segundo, las pruebas indicaron que no hubo pérdida de paquetes, esto demuestra que la pila de protocolos TCP/IP en el núcleo de Mininet, es bastante sólida y que el controlador Ryu mantiene el control de los flujos incluso situaciones de alta demanda.

- **Sustitución de hardware de capa 3:** Este entorno ha probado su capacidad para emular las funciones de conmutadores de capa 3 y dispositivos industriales que costarían una gran suma. Al utilizar hardware reutilizado de años anteriores, se ha creado un entorno educativo que no tiene nada que envidiar a la infraestructura moderna de hardware propietario.

4.3.4 Monitoreo de saturación y control en tiempo real mediante SDN

La verdadera ventaja de la arquitectura implantada no solo está en la velocidad, sino también en la completa visibilidad que el controlador Ryu tiene sobre el plano de datos. A continuación, se muestra como el enfoque SDN elimina la falta de claridad de las redes tradicionales:

- **Visibilidad centralizada:** A diferencia de lo que ocurre en una red normal, donde el tráfico permanece oculto A menos que se instalen sensores o se configure manualmente el protocolo SNMP en cada dispositivo, el controlador Ryu recibe datos periódicos de cada puerto del switch virtual a través de mensajes. Esto permite que al momento de las prácticas se observe en tiempo real la cantidad de bytes que atraviesan un enlace específico dentro de la topología de árbol.
- **Capacidad de respuesta y calidad de servicio:** se ha verificado que al identificar niveles de saturación A las pruebas realizadas con Iperf, el controlador puede programarse para implementar políticas de calidad del servicio o desviar flujos de manera inmediata. Esta funcionalidad es importante para que los estudiantes, al momento de realizar prácticas, comprendan la diferencia operativa entre el "reenvío ciego" de las redes tradicionales y el "control inteligente" de una red programable.

4.4 ENTORNO VIRTUAL VS LABORATORIO FÍSICO

Finalmente, se presenta el análisis comparativo entre el modelo de enseñanza tradicional que utiliza hardware y la opción virtual propuesta mediante SDN. La evaluación busca resaltar aspectos importantes como; la efectividad operativa de la infraestructura y la ventaja arquitectónica del software desarrollado en el estudio.

4.4.1 Validación final: Estación piloto vs. Hardware industrial físico

Todas las pruebas obtenidas en la estación de trabajo piloto que mantiene el procesador Intel Core i5-5200U valida que la emulación por software es completamente adecuada para lograr los objetivos educativos en la carrera de telecomunicaciones, la computadora ha demostrado ser capaz de replicar comportamientos de red complejos que hasta hace poco exigían un hardware especializado y costoso.

A través del uso de recursos computacionales disponibles y software libre (Mininet/Ryu), se logra emular funciones de switches de capa 3 y controladores industriales cuyo costeo en el mercado podría alcanzar miles de dólares en licencias y equipos físicos. Esto establece la implementación como una solución no solo técnica y funcionalmente sólida, sino que también como un modelo de infraestructura sostenible que permite a la facultad innovar sin depender de altos presupuestos para la compra de hardware de propiedad.

4.4.2 Evaluación de infraestructura y despliegue

La emulación en la estación de trabajo piloto, permite eliminar las limitaciones de costos y espacio que, a lo largo de la historia, han afectado a los laboratorios físicos. En una institución donde los recursos son altamente demandados, la virtualización surge como la única solución para asegurar que cada alumno tenga su propio "rack" de red sin la necesidad de gastar miles de dólares.

Tabla 9

Comparativa técnica de beneficios con respecto a la infraestructura

Criterio de evaluación	Laboratorio físico (Hardware)	Laboratorio virtual (Proyecto)	Impacto institucional
Costos de equipos	Alto: Inversión en dispositivos de red.	Inexistente: Uso de PCs del laboratorio o personales.	Ahorro presupuestario
Espacio requerido	Físico: Racks – mesas	Digital: Disco duro	Optimización de aulas.

Criterio de evaluación	Laboratorio físico (Hardware)	Laboratorio virtual (Proyecto)	Impacto institucional
Tiempo de Setup	Horas: Cableado manual	Segundos: Carga de script	Eficiencia académica.
Riesgo de daños	Alto: Desgaste de puertos	Nulo: Entorno de software	Sostenibilidad técnica
Escalabilidad	Limitada al inventario	Virtualmente ilimitada	Crecimiento dinámico

Nota. Se presenta el análisis comparativo entre la implementación de hardware físico para prácticas frente a un entorno virtualizado, la principal ventaja es el ahorro de dinero en los costos de equipos. *Fuente:* Autor.

Mientras un laboratorio físico necesita la compra constante en switches, routers y cableado que tienden a desgastarse con el tiempo, la infraestructura virtual aprovecha el hardware ya existente lo que lleva a que el costo de inversión sea cero:

- **Adaptabilidad de topologías:** en un entorno físico, para cambiar de una red en Estrella a una red de tipo árbol se requiere mover cables y reconfigurar puertos de manera manual, esa transición se logra en cuestión de segundos al ejecutar el script desarrollado "Lab_ucsg.py"
- **Escalabilidad:** agregar 10 hosts adicionales en un entorno físico significa comprar tarjetas de red y cables; en el entorno virtual solo se necesita actualizar una variable en el código.

4.4.3 Estructura de red: Modelo tradicional frente al modelo SDN

Para entender el valor de este proyecto, es muy importante definir por qué el laboratorio físico convencional se clasifica como un entorno tradicional; la principal diferencia se debe al lugar donde se localiza la “inteligencia de red:

- **Plano de control distribuido (convencional):** En los laboratorios tradicionales cada switch toma decisiones por su cuenta; esto significa que, al momento de realizar prácticas, la configuración de los equipos tiene que ser uno a uno, lo que hace difícil tener una visión de la red y

el aprendizaje se basará en acciones repetitivas y no en la lógica del tráfico.

- **Centralización del control (SDN):** La solución utiliza el controlador Ryu, el cual centraliza toda la lógica; en las prácticas se interactúa con un sistema orquestado que permite una total programabilidad del tráfico, alineándose con las tendencias actuales de infraestructura como código (IaC).
- **Desacoplamiento de capas:** En el modelo SDN, el plano de control opera de manera independiente al plano de datos; esta configuración permite que se innove en protocolos sin el temor de dañar el hardware, creando un entorno de "sandbox" o caja de arena ideal para la experimentación.

Tabla 10

Comparativa crítica de arquitecturas

Criterio	Red tradicional (Hardware)	Red SDN (Proyecto)	Ventaja técnica
Control	Distribuido (en cada equipo)	Centralizado (Controlador Ryu)	Mayor visibilidad y gestión global
Configuración	Manual y propensa a errores	Programable (Lenguaje Python)	Automatización y rapidez de despliegue.
Visibilidad	Opaca (Requiere sensores)	Transparente (Monitoreo en Ryu)	Facilita la auditoría de protocolos.

Criterio	Red tradicional (Hardware)	Red SDN (Proyecto)	Ventaja técnica
Flexibilidad	Estática (Física)	Dinámica (Lógica / Scripts)	Permite múltiples topologías simultaneas.
Innovación	Limitada por el fabricante	Abierta (OpenFlow 1.3)	Libertad para programar nuevos servicios.

Nota. En la tabla se destaca la transición de un control manual y estático basado en hardware hacia una gestión centralizada, programable y de código abierto (OpenFlow), que optimiza la visibilidad y la flexibilidad operativa de la red. *Fuente:* Autor.

CAPÍTULO 5
CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- Se logró crear una estructura de red dual con las topologías estrella y árbol, que cumple con las necesidades técnicas de la estación piloto para la implementación en el laboratorio de telecomunicaciones. La ejecución evidenció que emplear scripts en Python proporciona una escalabilidad casi infinita, facilitando la transición de una configuración con cuatro hosts a una estructura jerárquica compleja en cuestión de segundos. Este diseño demuestra que la división lógica de subredes (10.0.1.x y 10.0.2.x) es el método más eficaz para organizar laboratorios educativos sin requerir hardware de enrutamiento físico adicional.
- La configuración del controlador Ryu junto con el emulador Mininet bajo el protocolo OpenFlow 1.3 resulta ser una solución muy estable. Se establece que la separación de los planos de red permite una gestión mejorada; el sistema mostró la capacidad de manejar eventos de Packet-In e introducir reglas de flujo con una precisión del 100%. Esta centralización de la inteligencia de red simplifica la tarea de configurar dispositivos de forma individual, convirtiendo la administración de la red en una actividad basada en programación lógica.
- Se verifica que el procesador Intel Core i5-5200U, aunque es un hardware de generaciones anteriores, tiene la capacidad plena para manejar un laboratorio de redes de última generación. Las pruebas de rendimiento con Iperf mostraron que la virtualización ligera de Mininet permite alcanzar tasas de transferencias de Gbits/s con una carga de CPU por debajo del 60%. Se concluye que la emulación por software elimina los cuellos de botella físicos de las tarjetas de red tradicionales, permitiendo que el laboratorio de telecomunicaciones aprovecha el máximo sus recursos tecnológicos actuales sin necesidad de gastar el nuevo equipamiento.
- La adopción del estándar OVF (.OVA) asegura que el laboratorio desarrollado sea una solución a la mano para la facultad, el procedimiento del empaquetado, qué incluyó la reinicialización de direcciones MAC y la creación de firmas digitales SHA-256, garantiza que el entorno de práctica

sea portátil, íntegro y estable en cualquier estación de trabajo que use VirtualBox. Este modelo de distribución elimina los tiempos dedicados a configuraciones manuales y la posibilidad de errores en la instalación, permitiendo una adopción masiva inmediata en el laboratorio de telecomunicaciones.

5.2 RECOMENDACIONES

- La adopción del estándar OVF (.OVA) asegura que el laboratorio desarrollado sea una solución a la mano para la facultad, el procedimiento del empaquetado, qué incluyó la reinicialización de direcciones MAC y la creación de firmas digitales SHA-256, garantiza que el entorno de práctica sea portátil, íntegro y estable en cualquier estación de trabajo que use virtual box. Este modelo de distribución elimina los tiempos dedicados a configuraciones manuales y la posibilidad de errores en la instalación, permitiendo una adopción masiva inmediata en el laboratorio de telecomunicaciones.
- Aunque este proyecto se enfocó en configuraciones de la topología estrella y árbol, se sugiere expandir el laboratorio a diseños de malla o arquitecturas con conexiones redundantes. Dado que mi internet permite la formación de bucles físicos, los estudiantes podrían investigar de qué manera el controlador SDN mejora la prevención de tormentas de difusión sin depender únicamente del protocolo Spaning Tree Tradicional, analizando algoritmos de enrutamiento más eficaces como “Dijksdtra” para determinar rutas con menor latencia.
- Para para fortalecer el aprendizaje, se recomienda incorporar herramientas de visualización gráficas en el sistema. La utilización de la interfaz web nativa de Ryu o la colaboración con plataformas como InfluxDB y Grafana permitiría a los estudiantes visualizar gráficos del uso del ancho de banda y la latencia de manera profesional. Esa representación visual ayudaría a interpretar los resultados obtenidos con Iperf, transformando los datos sin procesar de la terminal, en información útil y fácil de analizar para propósitos académicos.
- Se sugiere mantener un chequeo semestral del archivo .OVA para asegurar que tanto el kernel de Ubuntu como las bibliotecas de Ryu y

Mininet se conserven en versiones estables y con las últimas actualizaciones de seguridad. Igualmente, es recomendable fomentar el uso de este laboratorio en formatos de aprendizaje híbrido, permitiendo que los estudiantes descarguen el entorno en sus computadoras personales para promover la práctica autónoma fuera del horario de clases.

BIBLIOGRAFÍA

Alfaify, L., Alnajem, N., Alanzi, H., Almutiri, R., Alotaibi, A., Alhazri, N., & Alqahtani, A. (2023). Performance evaluation of SDN controllers: RYU and POX for WBAN-based healthcare applications. *2023 International Conference on modeling & e-information research, artificial learning and digital applications (ICMERALDA).* Karawang, Indonesia: IEEE. https://doi.org/10.1109/ICMERALDA60125.2023.10458183

Amaya Fariño, L. M. (2022). SDN Redes definidas por Software usando MiniNet. *Revista Científica Y Tecnológica UPSE, 9*(1), 48-56. https://doi.org/10.26423/rctu.v9i1.489

Ariganello, E. (2020). *Redes Cisco: Guía de estudio para la certificación CCNA 200-301*. RA-MA Editorial. https://elibro-net.ucsg.idm.oclc.org/es/ereader/ucsg/222695

Askar, S., & Keti, F. (2021). Performance evaluation of different SDN controllers: A review. *International journal of science and business: IJSB, 5*(6), 67-80. https://doi.org/10.5281/zenodo.4742771

Blanchet, R., Pérez, S. C., & Facchini, H. A. (2021). *Estudio y simulación de redes definidas por software y automatización de red.* Red de Universidades con Carreras en Informática.

Blanco Torres, A. S. (09 de Marzo de 2024). *Sistemas operativos*. Sistemas operativos. https://sistemas-operativos180.webnode.com.co/l/este-es-un-articulo-con-imagenes2/

Casilimas Fajardo, C. A., & Cáceres Guevara, J. E. (2022). *Arquitectura y funcionamiento de redes definidas por software (SDN).* Universidad Distrital Francisco Jose de Caldas.

Estévez Almeida, S. I. (2022). *Implementación de una infraestructura como servicio para desarrollo de prácticas SDN y NFV.* Universidad Técnica del Norte.

Forouzan, B. A. (2021). *Data communications and networking with TCP/IP Protocol Suite* (6ª ed.). McGraw-Hill Education.

González, C., Flauzac, O., & Nolot, F. (2020). Diseño de un entorno de pruebas SDN para soportar el IdC: prototipo y evaluación. *I+D Tecnológico (Universidad Tecnológica de Panamá), 16*(1), 69-77. https://doi.org/10.33412/idt.v16.1.2445

Jácome Paredes, D. A. (2025). Redes definidas por software y su impacto en el uso de aplicaciones de video. *Ciencia y Educación, 6*(1), 71-81. https://doi.org/10.5281/zenodo.16509716

Keerthana, B., Balachandra, M., Hebbar, H., & Muniyal, B. (2022). Performance comparison of various controllers in different SDN topologies. In Springer, *Expert clouds and applications* (pp. 297-309). Springer Singapore.

Kurose, J., & Ross, K. W. (2021). *Computer networking: A top-down approach* (8VA ed.). Pearson.

Mahmoud Huda, N. A. (2026). Comparative Performance Evaluation of Ryu and OpenDaylight SDN Controllers Using Mininet. *AlQalam Journal of Medical and Applied Sciences, 9*(1), 64-67. https://doi.org/10.54361/ajmas.269112

Mediero Martí, C. (2021). *Análisis de redes definidas por software (SDN) y su aplicación en el entorno sanitario.* Universitat Oberta de Catalunya (UOC). https://hdl.handle.net/10609/126808

Menéndez Regalado, E. A., & Vera Rendón, E. J. (2023). *Mapeo sistemático sobre redes definidas por software con virtualización de las funciones de red.* Universidad Politécnica Salesiana.

Microinstalaciones. (09 de Febrero de 2021). *MICROINSTALACIONES*. MICROINSTALACIONES. https://microinstalaciones.com.ar/cual-es-la-diferencia-entre-una-lan-y-una-wan/

Mohammad Nowsin, A. S., I-Shyan , H., Muhammad Saibtain, R., & Mohammad Syuhaimi, A.-R. (2024). Una evaluación cualitativa y comparativa del

rendimiento de los controladores SDN lógicamente centralizados a través del emulador Mininet. *Computers (MDPI), 13*(4), 85. https://doi.org/10.3390/computers13040085

Munusamy, R., & Shukla, S. (2024). *Performance comparison of SDN controllers In different network topologies.* Research Square.

Open Networking Foundation. (06 de Enero de 2026). *ONF*. Open Networking Foundation. https://opennetworking.org/sdn-definition/

Paucar Asqui, A. P. (2022). *Análisis e implementación de un prototipo de red SDN en el campus norte de la UNACH.* Universidad Nacional de Chimborazo.

Peterson, L. L., & Davie, B. S. (2021). *Computer Networks: A Systems Approach* (6th ed.). Morgan Kaufmann. https://book.systemsapproach.org/

ProcessOn. (20 de Junio de 2025). *ProcessOn*. ProcessOn. https://www.processon.io/es/blog/star-network-topology-layout

Santamaría Sandoval, J. R. (2020). La gestión en redes definidas por software (SDN) desde la perspectiva de FCAPS. *Repertorio Científico, 23*(2), 1-12. https://doi.org/10.22458/rc.v23i2.2870

Silva, J. (2021). Tecnología de red definida por software para el aprendizaje en grupos de investigación y educación. *Revista Innova Educación, 3*(3), 85-96. https://doi.org/10.35622/j.rie.2021.03.005

Tanenbaum, A. S., & Wetherall, D. J. (2021). *Computer networks (Redes de computadoras)* (6ta ed.). Pearson.

Toledo Gómez, B. D. (2022). *Prototipo de red definida por software usando protocolo OpenFlow para optimizar la red existente en la carrera de Tecnologías de la Información.* Universidad Estatal del Sur de Manabí.

Torres Quijije, Á. I., & Zuñiga Paredes, A. R. (2021). Análisis de herramientas que permitan el modelado de tráfico en redes SD. *CentroSur Editorial, 4*(2), 55-68. https://doi.org/10.37955/cs.v4i2.60

Vega Guallpa, A. J., Andrade Cárdenas, D. P., & Pinos Castillo, L. F. (2022). Análisis comparativo de infraestructuras de redes SDN y redes tradicionales IP. *Pro Sciences Journal*, *6*(46), 48-58. https://doi.org/10.29018/issn.2588-1000vol6iss43.2022pp71-82

Zúñiga Sánchez, M. A., Móntece Moreno, O. R., & Posligua Ruiz, G. E. (2025). Impacto de las redes definidas por software (SDN) en la gestión de redes tradicionales mediante un enfoque de simulación y aprendizaje. *Revista Synergía*, *4*(1), 171-186. https://doi.org/10.48204/synergia.v4n1.7182

ANEXOS

Manual de implementación / Laboratorio virtual de redes SDN

Para asegurar que el laboratorio virtualizado logre su objetivo educativo en el laboratorio de telecomunicaciones de la Facultad de Educación Técnica para el Desarrollo de la UCSG, se ha elaborado un manual de implementación paso a paso. Este documento está diseñado para guiar a los estudiantes desde la instalación del software fundamental hasta una evaluación crítica de los protocolos de red, garantizando una experiencia de aprendizaje continua, incluso para quienes no tienen conocimientos previos en sistemas Linux o Redes Definidas por Software.

PASO 1: Preparación del entorno host (anfitrión) / Instalación básica

El primer paso para cualquier estudiante o docente es preparar la computadora (sistema anfitrión) donde se realizará la implementación, con el fin de que pueda soportar la virtualización.

Descarga del hipervisor

1. Abrir un navegador en la web y acceder a la página oficial de Oracle VM VirtualBox.
2. Descargar el instalador adecuado para el sistema operativo anfitrión (por ejemplo, Windows hosts). Se sugiere usar la versión 6.1 o más reciente.

Instalación de VirtualBox:

1. Ejecutar el archivo descargado.
2. Seguir el asistente de instalación estándar, haciendo clic en "Siguiente" en cada pantalla. No es necesario alterar las configuraciones de red predeterminadas del instalador.
3. Completar la instalación y abrir el programa VirtualBox.

PASO 2: Configuración de la Máquina Virtual (Dos Opciones)

Para comenzar a usar el entorno Ubuntu, el estudiante puede encontrarse con dos posibilidades, dependiendo de si el instructor le ha proporcionado el archivo empaquetado del proyecto o si necesita crearlo él mismo.

Escenario A: Creación desde Cero (Sin archivo .OVA)

Si al ejecutar la instalación no se tiene el archivo preconfigurado, tendrá que instalar el sistema y las herramientas manualmente:

1. Creación de la Máquina Virtual: En VirtualBox, hacer clic en "Nueva". Asignar el nombre "Ejemplo: Ubuntu SDN", seleccionar tipo "Linux" y versión "Ubuntu (64-bit)".
2. Asignación de Recursos: En el asistente, asignar 4096 MB (4 GB) de memoria RAM y 2 CPUs. Crear un disco duro virtual de por lo menos 25 GB.
3. Instalación del sistema operativo: Configurar la imagen ISO de Ubuntu 22.04 LTS en la unidad virtual y encender la máquina. Seguir las instrucciones del asistente de Ubuntu para llevar a cabo una instalación estándar del sistema operativo.
4. Instalación de Dependencias SDN: Al acceder al escritorio de Ubuntu, abrir una terminal utilizando "Ctrl + Alt + T" y ejecutar los comandos proporcionados para obtener el software necesario:
   sudo apt update
   sudo apt install python3-pip -y
   pip3 install ryu
5. Creación del Script: En la misma terminal, generar el archivo correspondiente a la topología:
   nano lab_ucsg.py

(En este momento, el estudiante debe insertar el código Python de la topología que fue creado en la Fase 2 del proyecto, guardar el archivo con Ctrl+O y salir con Ctrl+X).

El código de la topología que se desarrolló, que incluye los switches s10, s20, s30 y los enlaces TCLink. Las 2 topologías se ejecutaban bajo el código:

```
from mininet.topo import Topo


class UCSG_Lab(Topo):

    def build(self):
```

```
# --- ESCENARIO 1: ESTRELLA (Tabla 7) ---
s1 = self.addSwitch('s1', dpid='1')
for i in range(1, 5):
    h = self.addHost('h%s' % i, ip='10.0.1.%s/24' % i)
    self.addLink(h, s1)

# --- ESCENARIO 2: ÁRBOL (Tabla 8) ---
# Switches: Core (s10) y Agregación (s20, s30)
s10 = self.addSwitch('s10', dpid='10')
s20 = self.addSwitch('s20', dpid='20')
s30 = self.addSwitch('s30', dpid='30')

# Uplinks entre switches (Troncales)
self.addLink(s10, s20)
self.addLink(s10, s30)

# Hosts de la Rama Izquierda (s20) - IPs según Tabla 8
h5 = self.addHost('h5', ip='10.0.2.1/24')
h6 = self.addHost('h6', ip='10.0.2.2/24')
self.addLink(h5, s20)
self.addLink(h6, s20)

# Hosts de la Rama Derecha (s30) - IPs según Tabla 8
h7 = self.addHost('h7', ip='10.0.2.3/24')
```

```
        h8 = self.addHost('h8', ip='10.0.2.4/24')
        self.addLink(h7, s30)
        self.addLink(h8, s30)

topos = { 'lab_ucsg': ( lambda: UCSG_Lab() ) }
```

Escenario B: Importación Rápida (Usando el archivo .OVA proporcionado por la FETD)

Si la facultad ya ha proporcionado el laboratorio empaquetado, el procedimiento se simplifica en tres pasos:

1. En VirtualBox, seleccionar el menú Archivo y luego Importar servicio virtualizado.
2. Localizar y elegir el archivo Laboratorio_SDN_UCSG. ova .
3. En la ventana de configuración, buscar la opción "Política de direcciones MAC" y optar por "Generar nuevas direcciones MAC para todos los adaptadores de red". Luego, hacer clic en Importar.

PASO 3: Guía Práctica Oficial (Ejecución en Ubuntu)

Una vez que el estudiante está dentro de Ubuntu (ya sea usando el Escenario A o B), el entorno se encuentra preparado. Desde este punto, se aplicará el Diseño Instruccional para desarrollar la práctica.

Identificación de la Práctica:

Título: Práctica 1 - Configuración de Flujos y Análisis de Latencia en Topología Jerárquica .

- **Duración estimada:** 45 minutos.
- **Nivel de dificultad:** Intermedia
- **Fundamentación Técnica:** En una red convencional, los switches aprenden automáticamente las direcciones MAC mediante el método de inundación (flooding). Durante esta práctica, se investigará cómo el modelo SDN asume esta responsabilidad. Se comprobará que el Switch OpenFlow, al recibir el primer paquete (ICMP/ARP), se

encuentra sin instrucciones y consulta al controlador Ryu mediante un mensaje Packet-In, provocando un retraso que se puede medir.

Equipamiento Virtual:

- Sistema Operativo: Ubuntu 22 .04 LTS (en un entorno virtual).
- Emulador: Mininet .
- Controlador: Ryu (conforme al estándar OpenFlow 1 .3).
- Archivo de despliegue: lab_ucsg .py .

Procedimiento Experimental:

1. ***Fase de Despliegue (El "Cerebro" y el "Cuerpo")***

El entorno SDN requiere la separación del plano de control del plano de datos.

Paso 1.1: Abrir una terminal inicial y activar el controlador Ryu ejecutando la aplicación de conmutación básica:

ryu-manager ryu.app.simple_switch_13

(La terminal mostrará un estado de espera, lo que indicará que el controlador está a la escucha en el puerto 6653).

Paso 1.2: Abrir una segunda terminal e iniciar la estructura de red mediante el script:

sudo python3 lab_ucsg.py

La consola cambiará al prompt "mininet>", indicando que los switches y hosts están activos. Después de eso, cuando se quiera iniciar nuevamente a la topología desarrollada, se puede iniciar directamente con el comando:

sudo mn --custom lab_ucsg.py --topo lab_ucsg -- controller remote,ip=127.0.0.1 -- switch ovs, protocols=OpenFlow13

Automáticamente nos llevará al prompt de mininet>

2. Etapa de Prueba (Análisis de la Red)

Paso 2.1: En la interfaz de Mininet, comprobar la conectividad total obligando a todos los nodos a intercambiar información:

mininet> pingall

Paso 2.2: Llevar a cabo una evaluación de latencia entre dos computadores específicos (por ejemplo, h1 y h4):

mininet> h1 ping -c 4 h4

3. Etapa de Inspección (Revisión de Protocolos)

Paso 3.1: Iniciar una tercera ventana en Ubuntu y poner en marcha el analizador de paquetes:

sudo wireshark

Paso 3.2: Escoger la interfaz virtual de bucle "lo" (Loopback) y aplicar el filtro openflow_v4 en la parte superior. Repetir el comando de ping en Mininet y observar cómo Wireshark registra el proceso de establecimiento de conexión (Handshake) y la configuración de reglas (Flow-Mod).

DECLARACIÓN Y AUTORIZACIÓN

Yo**, Pincay Portilla, Joselin Julexy**, con C.C: **095305474-9** autora del trabajo de titulación: **Diseño e implementación de un entorno virtual de laboratorio para la realización de prácticas de redes LAN en el área de telecomunicaciones,** previo a la obtención del título de **INGENIERÍA EN TELECOMUNICACIONES** en la Universidad Católica de Santiago de Guayaquil.

1.- Declaro tener pleno conocimiento de la obligación que tienen las instituciones de educación superior, de conformidad con el Artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de titulación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la SENESCYT a tener una copia del referido trabajo de titulación, con el propósito de generar un repositorio que democratice la información, respetando las políticas de propiedad intelectual vigentes.

Guayaquil, 02 de marzo del 2026

f. _____

Nombre: **Pincay Portilla, Joselin Julexy**

C.C: **0953054749**

REPOSITORIO NACIONAL EN CIENCIA Y TECNOLOGÍA

FICHA DE REGISTRO DE TESIS/TRABAJO DE TITULACIÓN

TEMA Y SUBTEMA:	Diseño e Implementación de un entorno virtual de laboratorio para la realización de prácticas de Redes LAN en el área de Telecomunicaciones.		
AUTOR(ES)	Pincay Portilla, Joselin Julexy		
REVISOR(ES)/TUTOR(ES)	Ing. Zamora Cedeño, Néstor Armando, Mgs.		
INSTITUCIÓN:	Universidad Católica de Santiago de Guayaquil		
FACULTAD:	Facultad de Educación Técnica para el Desarrollo		
CARRERA:	Carrera de Telecomunicaciones		
TITULO OBTENIDO:	Ingeniero en Telecomunicaciones		
FECHA DE PUBLICACIÓN:	02 de marzo de 2026	**No. DE PÁGINAS:**	124
ÁREAS TEMÁTICAS:	Redes definidas por software (SDN), Laboratorio virtual, Tecnología educativa.		
PALABRAS CLAVES/ KEYWORDS:	Redes, SDN, Mininet, Ryu, Virtualización, Laboratorio, Topologías, Protocolos.		

RESUMEN/ABSTRACT: Esta tesis aborda el desafío de la brecha tecnológica y el costo de crear laboratorios de redes de datos físicos en la facultad de educación técnica para el desarrollo (FETD). El objetivo principal fue diseñar e implementar un entorno de laboratorio virtualizado, que opera bajo el paradigma de redes definidas por software (SDN), con el emulador Mininet y el controlador Ryu en el estándar OpenFlow 1.3.

Este enfoque implicó diseñar dos topologías fundamentales (Estrella y árbol jerárquico) y utilizar scripts de Python para automatizarlas, optimizando así el despliegue de una estación de trabajo piloto con recursos de hardware intermedios (Intel Core i5-5200U). Las pruebas de conectividad y rendimiento con herramientas, incluidas Iperf y Wireshark, mostraron una tasa de transferencia (orden de Gbits/segundo) y una latencia de procesamiento reactivo de menos de 1 ms: el uso de CPU fue inferior al 60% por lo que el sistema se mantuvo estable.

Finalmente, el laboratorio se ensambló en formato OVF (.OVA) que podría usarse rápidamente para desplegarlo en cualquier ubicación del campus. Se concluye que la solución propuesta no solo elimina las necesidades hardware propietario y costoso, sino que establece una plataforma pedagógica flexible y escalable que se ajusta a las tendencias actuales en infraestructura como código (IaC) y refuerza el proceso de enseñanza-aprendizaje en la carrera de telecomunicaciones.

ADJUNTO PDF:	☒ SI	☐ NO
CONTACTO CON AUTOR/ES:	**Teléfono:** +593-9982632505	**E-mail:** joselin.pincay@cu.ucsg.edu.ec
CONTACTO CON LA INSTITUCIÓN (C00RDINADOR DEL PROCESO UTE):	**Nombre:** Ing. Ubilla González, Ricardo Xavier, MsC	
	Teléfono: +593-999528515	
	E-mail: ricardo.ubilla@cu.ucsg.edu.ec	

SECCIÓN PARA USO DE BIBLIOTECA	
Nº. DE REGISTRO (en base a datos):	
Nº. DE CLASIFICACIÓN:	
DIRECCIÓN URL (tesis en la web):	