TecnoLógicas

Implementación de un dispositivo para pruebas de conectividad en una red definida por software sobre el protocolo IPv6

Implementation of a Device for Connectivity Testing of a Software-Defined Network Over the IPv6 Protocol

Jorge Enrique Herrera Rubio[1]; Luz María Ballesteros Gómez[1]

[1]Universidad de Pamplona, Pamplona-Colombia

Correspondencia: jherrera@unipamplona.edu.co

Recibido: 16 Abril 2025 **Aceptado:** 04 Noviembre 2025 **Disponible:** 30 Noviembre 2025

Cómo citar / How to cite
J. E. Herrera Rubio, and L. M. Ballesteros Gómez, "Implementación de un dispositivo para pruebas de conectividad en una red definida por software sobre el protocolo IPv6," *TecnoLógicas*, vol. 28, no. 64, e3459, 2025.
https://doi.org/10.22430/22565337.3459

Resumen

Con la ayuda de herramientas de tecnologías de virtualización se han desarrollado nuevas plataformas de redes de infraestructura más eficientes y flexibles, como las redes definidas por software (SDN, por sus siglas en inglés), una solución práctica al separar las funciones de los planos de datos y de control. El objetivo principal de la investigación fue implementar un banco de pruebas para evaluar el rendimiento y comportamiento de una red SDN ejecutándose sobre el protocolo IPv6. El alcance de la investigación abarcó desde el diseño hasta la puesta en marcha de un entorno en un laboratorio experimental; la metodología empleada consistió en la aplicación de la técnica Top-Down. Se estructuraron cinco fases: se identificaron los requisitos, se ejecutó el diseño conceptual, se estableció la ingeniería de detalle del subsistema, se integró y se validó el sistema con la puesta en marcha en un ambiente controlado. Los resultados obtenidos fueron: la correcta validación del protocolo IPv6, la garantía en la entrega de paquetes de texto e imágenes y la baja latencia del tráfico. Se concluye que con la validación del uso de redes SDN sobre IPv6 se obtienen ventajas claras en la flexibilidad en la infraestructura de red, administración centralizada del tráfico y eficacia en la gestión del tráfico. La interoperabilidad garantiza la compatibilidad con los diferentes tipos de hardware en el momento de la integración de dispositivos y plataformas respetando los estándares de la industria.

Palabras clave

Hardware, infraestructura, mininet, red de próxima generación, Ryu.

Abstract

With the help of virtualization technology tools, new, more efficient, and flexible infrastructure network platforms have been developed, such as software-defined networking (SDN) as a practical solution by separating the functions of the data and control planes. The main objective of the research was the implementation of a testbed to evaluate the performance and behavior of an SDN network running on the IPv6 protocol. The scope of the research ranged from the design to the implementation of an experimental laboratory environment. The methodology used consisted of applying the Top-Down technique. Five phases were structured: requirements were identified, the conceptual design was executed, detailed subsystem engineering was established, and the system was integrated and validated with implementation in a controlled environment. The results obtained were: successful validation of the IPv6 protocol, guaranteed delivery of text and image packets, and low traffic latency. Finally, it is concluded that validating the use of SDN networks over IPv6 provides clear advantages in terms of network infrastructure flexibility, centralized traffic management, and efficient traffic management. Interoperability ensures compatibility with different types of hardware when integrating devices and platforms, while respecting industry standards.

Keywords

Hardware, infraestructure, mininet, next-generation network, Ryu.

1. INTRODUCCIÓN

Una de las particularidades de las redes definidas por software (SDN, por sus siglas en inglés) [1] es la forma como han emergido dando apoyo a la *interworking* en la evolución y el desarrollo de las tecnologías de infraestructura (TI), lo que ha permitido redefinir la forma como se diseñan e implementan las redes. La flexibilidad en la programación de la SDN [2] está en la separación del plano de control del plano de datos que facilita la gestión y administración de las redes [3].

En comparación con las estructuras de redes actuales que utilizan excesivos equipos de hardware y software, las SDN posibilitan que los administradores de servicios de redes tomen decisiones inteligentes de enrutamiento y gestión del tráfico por medio de la abstracción de un controlador centralizado (por ejemplo: OpenDaylight, ONOS, Ryu, Floodlight, Trema, POX, Cisco APIC-EM y ONAP, entre otros) en la capa física [4].

Los equipos como switchet, router y puntos de acceso se convierten en dispositivos programables [5] que ejecutan las instrucciones del controlador central [6] para reducir el equipamiento de infraestructura. Sin embargo, este tipo de redes enfrentan algunos desafíos que enfrentan este tipo de redes está en: la compatibilidad, la migración con redes de próxima generación, la seguridad y las competencias laborales del personal técnico que administra la red [7].

Las redes SDN para optimizar y simplificar sus funciones se deben configurar en entornos virtuales y requieren para su implementación de dispositivos físicos, que ya no depende de la configuración manual y estática de cada dispositivo; para ello introduce un controlador centralizado que ejecuta las instrucciones de manera dinámica para: automatizar procesos, dar eficiencia en operación y configuración [1], hacer visible la red, permitir flexibilidad y seguridad avanzada [8]; para ello se requiere de la instalación del controlador como también de seleccionar equipos compatibles con SDN, configurar los protocolos requeridos y ejecutar pruebas de funcionalidad [9].

El objetivo de esta investigación fue evaluar el rendimiento de SDN, mediante un entorno experimental que permita analizar la eficiencia en la gestión del tráfico, la asignación de recursos y la latencia en distintos escenarios de red.

Las redes SDN se adaptan a las redes convergentes fácilmente con el protocolo IPv6 [10] para mayor seguridad mediante mecanismos de OpenFlow [4] permitiendo que los dispositivos de red y el controlador reconozcan los campos de la cabecera de este protocolo entre los dispositivos de red y el controlador para garantizar la integridad en el envío de los datos [11].

La creciente demanda de servicios de datos en tiempo real exige infraestructuras de cloud computing y redes móviles capaces de garantizar una calidad de servicio (QoS) óptima y el ancho de banda necesario para su prestación del servicio [9].

Frente a estos desafíos, las redes SDN surgen como una solución viable. Su principio fundamental se basa en la disociación entre el plano de control y el plano de datos que permite procesos de gestión ágiles, tal y como demuestran [12]. Al desvincular el control de la infraestructura física subyacente, aborda efectivamente los problemas de escalabilidad en entornos masivos como los centros de datos [3], a la vez que proporciona la agilidad necesaria para reconfigurar dinámicamente los servicios y la topología de la red en tiempo real.

La orquestación de SDN es la solución clave para integrar redes fragmentadas a nivel nacional o regional como exponen [13], a través de la gestión de múltiples controladores de forma coordinada para aprovisionar servicios en entornos geográficamente dispersos y heterogéneos; en donde existen diferentes proveedores y multiplicidad de equipos de red en escenarios complejos [14].

De igual forma, el análisis planteado por [15] identifica tipos de red TCP/IP y selecciona protocolos, controladores y herramientas de emulación para la implementación de redes combinadas o híbridas buscando medir el rendimiento con respecto a las redes SDN.

En ambientes corporativos de redes pequeñas y medianas [10] proponen el despliegue y la implementación de redes SDN, centrando el estudio en la problemática de la ubicación de controlador de procesamiento paralelo (CPP) y controlador de procesamiento paralelo múltiple (MCPP) como un desafío técnico por medio de la revisión de las métricas de latencia y las técnicas de equilibrio de carga del controlador del sistema.

En su estudio, [15] se enfocan en OpenFlow para determinar el rendimiento de varios SDN utilizando Mininet de acuerdo a sus capacidades; utilizan seis métricas de rendimiento: ancho de banda, tiempo de ida y vuelta, retraso, fluctuación, pérdida de paquetes y rendimiento. Comprueban que el controlador Ryu supera al controlador OpenDayLight en términos de latencia, pérdida de paquetes y tiempo de ida y vuelta, pero este último supera al Ryu en términos de rendimiento, ancho de banda y *jitter*.

En lo que respecta a la seguridad en SDN, [16] utilizan el mecanismo de autenticación basado en la seguridad de la comunicación local de enlaces utilizando el Protocolo de Internet Versión 6 (IPv6) a través de OpenFlow; por su parte, [17] proponen un mecanismo de autenticación basado en SDN para verificar la identidad de los paquetes Protocolo de descubrimiento de vecinos (NDP) transmitidos en una LAN por medio del control centralizado y la programación de los procesos.

En la investigación de [18] se aborda el aprendizaje automático (ML) sobre las soluciones, los parámetros y los entornos de evaluación enfocados en mejorar la seguridad y la inteligencia en redes SDN. Los autores [19] detallan los esfuerzos recientes para incluir la inteligencia artificial (IA) en tres subcampos: aprendizaje automático, metaheurística y sistemas de inferencia difusa para su aplicación y uso potencial en el paradigma SDN.

Los estudios previos que aportaron al objeto de estudio se sintetizan en la Tabla 1 resaltando las variables por temas de afinidad a la investigación desarrollada.

Basado en lo anterior, el objetivo de la investigación consistió en implementar un dispositivo para pruebas de conectividad de una red definida por software sobre el protocolo IPv6 con el fin de comprobar la interoperabilidad y compatibilidad con los diferentes tipos de hardware.

Tabla 1. Síntesis metodológica y hallazgos en la investigación. Fuente: elaboración propia.

Tema	Autores	Variables de estudio	Método de estudio	Técnicas empleadas	Resultados obtenidos	Conclusiones principales
Arquitectura SDN y controladores	[5]	Rendimiento (latencia, throughput).	Revisión sistemática [5, 10], Experimentación en emuladores (Mininet) [6].	Análisis comparativo de arquitecturas.	ONOS y OpenDaylight superan en escalabilidad a Ryu y POX [6].	La elección y ubicación del controlador para el rendimiento en redes a gran escala. Los controladores distribuidos son esenciales para entornos de producción.
	[6]	Escalabilidad y confiabilidad.		Emulación de topologías de red.	La latencia y la resiliencia dependen críticamente de la solución al CPP [10].	
	[10]	Ubicación óptima de controladores (CPP).		Algoritmos heurísticos para CPP.		
Seguridad en SDN	[17]	Tasa de detección de ataques (SLAAC).	Diseño e implementación de mecanismos [19], revisión de literatura [19].	Mecanismos de detección de ataques SLAAC.	"SADetection" para identificar y mitigar ataques de suplantación en IPv6 [17].	La implementación de defensas proactivas y específicas, para IPv6, y la IA en automatización de ciberseguridad.
	[19]	Eficacia de protocolos de autenticación.		Frameworks de IA/ML para IDS.	La IA permite una respuesta dinámica y adaptativa a las amenazas [19].	
Integración SDN/IPv6 y casos de uso	[12]	Interoperabilidad en entornos heterogéneos.	Estudio de caso [12], modelado y simulación [13].	Análisis de idoneidad de controladores.	ONOS y ODL son adecuados para teleprotección en redes eléctricas [12].	SDN actúa como un facilitador clave para la integración de IPv6 y la gestión de redes de próxima generación en sectores críticos.
	[13]	Rendimiento en aplicaciones críticas (redes eléctricas, QKD).		Protocolos de orquestación para QKD.	SDN es viable para orquestar la gestión de claves en redes QKD [13].	
IA y automatización en SDN	[18]	Precisión en clasificación de tráfico y detección de anomalías.	Revisión exhaustiva [18], [3].	Algoritmos de ML para optimización de red.	ML/DL mejora la detección de intrusos y la eficiencia del tráfico [18].	La sinergia SDN-IA es fundamental para crear redes autónomas, seguras y auto-optimizadas.
	[3]	Eficiencia en el balanceo de carga.		Estrategias de balanceo de carga.	El balanceo de carga inteligente mejora la tolerancia a fallos [3].	

2. MATERIALES Y MÉTODOS

La metodología utilizada en la investigación es de tipo experimental [20] que se aplica sobre una red de laboratorio en un ambiente controlado para emular la implementación de la red SDN [5] sobre protocolo IPv6 para pruebas de conectividad, como se muestra en la Figura 1 [21].

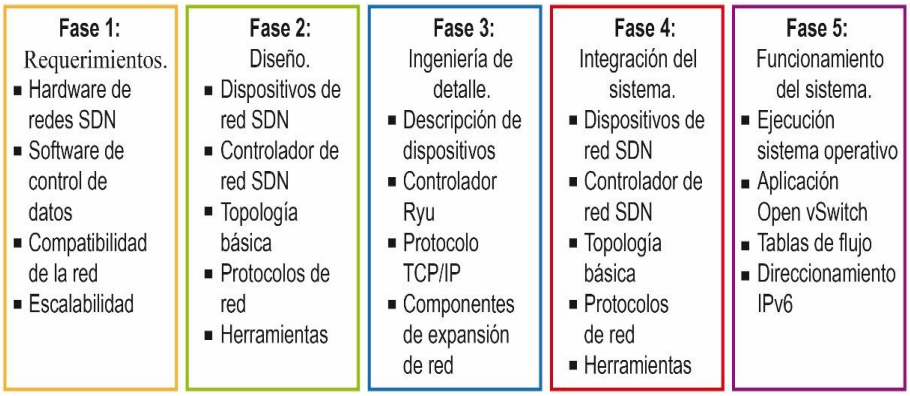

Figura 1. Estructura de la metodología Top Down. Fuente: elaboración propia adaptada de [7].

En la fase 1, enfocada en la identificación de los requerimientos generales del sistema, se determinan los requisitos esenciales de una red SDN. En la fase 2 se realiza el diseño global del sistema, mediante la subdivisión del sistema en cinco módulos: dispositivos de red, diseño del controlador de red, topologías básicas de la red, selección de los protocolos y las herramientas de expansión de la red. Para la fase 3, se considera la ingeniería de detalle; en la fase 4 se realiza la consolidación de todos los módulos y submódulos y, finalmente, en la fase 5 se ejecutan la implementación y las pruebas de funcionamiento [1].

2.1 Fase 1: identificación de los requerimientos del sistema

El proceso se enfoca en comprender y documentar las necesidades y especificaciones técnicas de la red. Sus componentes son: a) hardware especializado en redes SDN; b) software de control de datos, núcleo central de la red que permite la configuración de políticas de red, protocolos de comunicación y estrategias de gestión de la red; c) compatibilidad de la red, proceso que facilita la gestión y administración de IPv6 como protocolo subyacente al hardware de SND y d) escalabilidad de la red SDN para implementar e incorporar más dispositivos a la infraestructura propuesta.

2.2 Fase 2: diseño global del sistema

Integra los elementos para estructurar el sistema y cumplir con los requerimientos por medio de la metodología Top-Down [22], a través de la fragmentación por módulos que engloban aspectos esenciales sobre el protocolo IPv6, con un enfoque metódico y detallado en el diseño.

La infraestructura base del proyecto a implementar la componen: a) dispositivos de red: se configuran los conmutadores y enrutadores SDN, equipos de cómputo (host) y otros componentes para facilitar la comunicación y el flujo de datos, que se gestionan de manera centralizada por software Mininet [5]; b) controlador de red SDN: se utiliza el Ryu [2] como núcleo central de la red SDN; c) topología básica de red, se adaptó la de tipo de red estrella jerárquica, d) protocolos de red: se configuran los protocolos para enrutamiento y enrutados para mayor seguridad; e) herramientas de expansión de la red: se utiliza el Flow Manager y Sistema operativo Armbian OS [23].

2.3 Fase 3: Ingeniería de detalle

En el contexto de la metodología Top-Down, los módulos globales diseñados inicialmente se dividen en submódulos con el fin de analizar y desarrollar de forma sistemática los componentes y desafíos del proyecto, como se describe a continuación:

a) Dispositivos de red SDN. -Host: dispositivos finales que generan el flujo de datos y alojan las aplicaciones; incluyendo la participación como víctimas y atacantes en las pruebas de seguridad realizadas con stegomalware y la exfiltración de datos, aquí se incluyen los equipos portátiles, computadores de escritorio. -Nodos de red SDN: incluyen los enrutadores y conmutador SDN [24].
b) Controlador de red SDN: ejecuta la gestión y el direccionamiento del tráfico. -El software de control de red SDN lo constituye el controlador Ryu, que se integra a la arquitectura ARM por su capacidad de procesamiento y bajo consumo de energía en una tarjeta Orange PI Lite 2 [12].
c) Topología básica de la red: es de estructura de árbol con un nodo raíz que se ramifica en subredes secundarias para organizar y el enrutar tráfico.
d) Protocolos de red: en la capa de aplicación se utilizan los protocolos: DNS, DHCP, SMTP, FTP, HTTP; en la capa de transporte, los protocolos UDP y TCP; en la capa de internet, el enrutamiento se direcciona por medio de los protocolos: IP, ICMP y NAT y la capa de interfaz de red, que realizan las conexiones físicas con la ayuda de los protocolos ARP y Ethernet [25].
e) Herramientas de expansión de red: para facilitar la escalabilidad de la red se configuran: dispositivos de red emulados que son los host, conmutadores y las tarjetas de red para conectar los equipos externos y el módulo SDN (ver la Figura 2). El producto final del módulo se diseña con FreeCAD en 3D y se lleva a una máquina CNC para que sea terminado y cortado físicamente en material acrílico, el diseño se acompaña de una tapa móvil y otra fija donde internamente se alojan los módulos de expansión junto con la Orange Pi Lite 2 y la fuente de alimentación [26].

Figura 2. Prototipo SDN de pruebas. Fuente: elaboración propia.

2.4 Fase 4: Integración del sistema

Es la fusión física de los elementos de hardware que hacen parte de las fases del proceso de construcción modular, como se observa en la Figura 3. En la etapa de dispositivos de red se ha optado por emplear host emulados con sistemas operativos Linux, cada uno configurado con sus respectivas interfaces de Ethernet emuladas por medio del software Mininet, lo que garantiza un costo más económico en comparación con el uso de hardware físico [8].

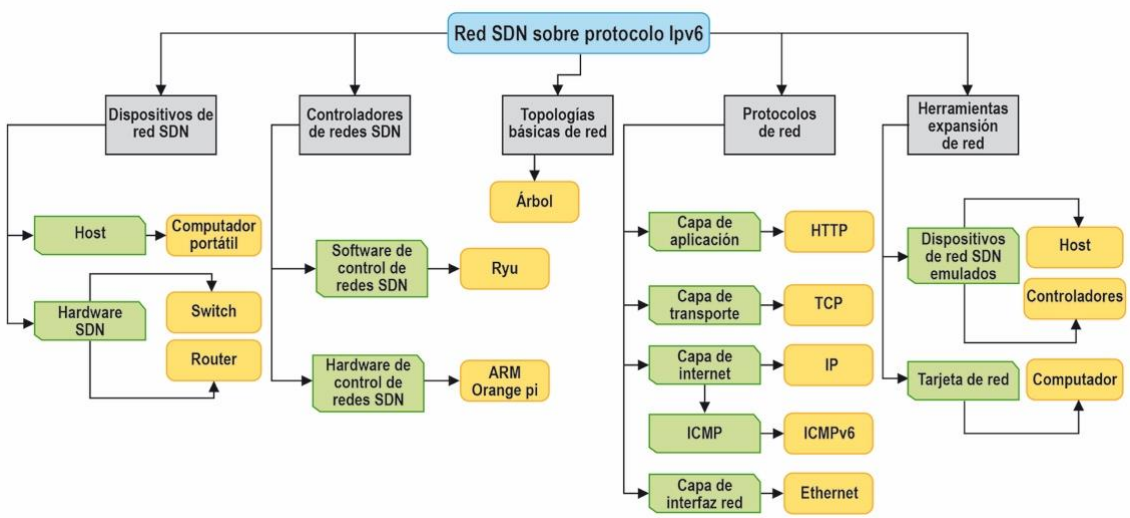

Figura 3. Diagrama Top-Down, diseño módulo del laboratorio de redes SDN. Fuente: elaboración propia.

En la etapa de topología tipo árbol se analiza el comportamiento de los paquetes en la red para garantizar el crecimiento de la misma con más equipos a futuro, como se observa en la Figura 4.

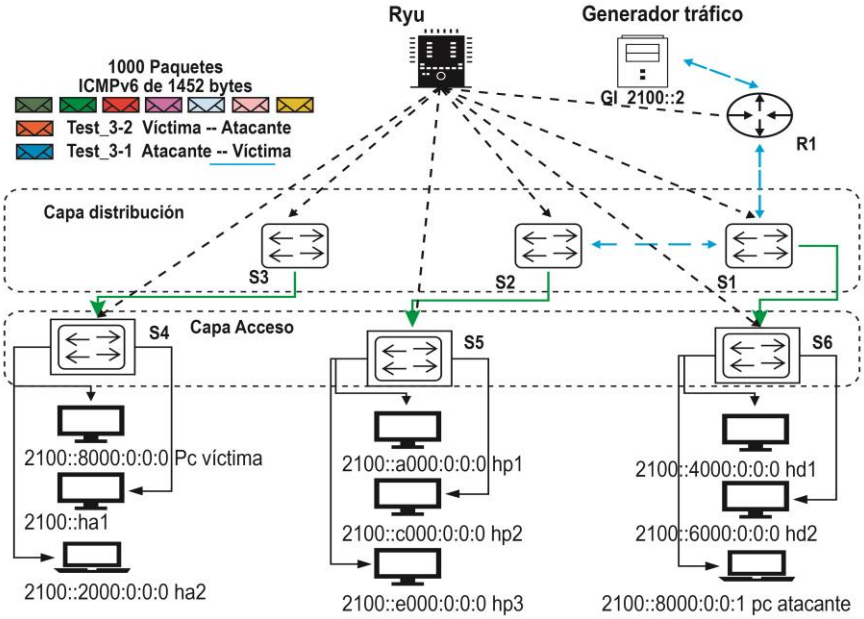

Figura 4. Rey SDN de pruebas implementada. Fuente: adaptado de [27].

Para la etapa de controladores de red, se consideran los requisitos necesarios en la ejecución del controlador Ryu [28], por medio de la inspección de hardware que cumplan con las condiciones técnicas establecidas, como se observa en la Tabla 2. Se seleccionó la plataforma Orange pi Lite 2 por su compatibilidad con el sistema operativo Armbian, prestaciones técnicas y facilidad en la programación del Ryu [29].

Tabla 2. Características de los candidatos de hardware. Fuente: elaboración propia.

Dispositivos de hardware para el laboratorio de redes SDN				
Nombres	Raspberry pi 3 B+	Mini PC	Raspberry pi 4	Orange pi lite 2
Procesador	Broadcom BCM2837B0	Intel Core I5-4258U	Broadcom BCM2711	H6 ARM Cortex™-A53
CPU	4 núcleos, 4 hilos, 1.4 GHz	2 núcleos, 4 hilos, 2.9 GHz	4 núcleos, 4 hilos, 1.5GHz	4 núcleos, 4 hilos, 1.8GHz
Memoria RAM	1GB LPDDR2	4GB DDR3	4GB LPDDR3	1 GB LPDDR3
Sistema operativo	Raspbian	Ubuntu 22.04.1 LTS	Raspbian	Armbian
Unidad de almacenamiento	32 GB (SDCard)	128 GB (SSD)	32GB (SDCard)	32GB (SDCard)
Controlador de SDN compatible	OpenDaylight, Ryu, POX	OpenDaylight, Floodlight, Beacon, Ryu, NOX, POX	OpenDaylight, Ryu, POX	OpenDaylight, Ryu, POX

La Orange Pi Lite 2 se soporta en un procesador H6 ARM Cortex™-A53 de 4 núcleos y 4 hilos que opera a 1.8GHz, con alto rendimiento y una memoria RAM de 1 GB LPDDR3 y la eficiencia energética para este tipo de investigaciones [14].

En la etapa de protocolos de red se adoptó el protocolo Hypertext Transfer Protocol (HTTP) para la ejecución de Flow Manager [30], con el fin de administrar los dispositivos y el controlador SDN; el protocolo Transmission Control Protocol (TCP) para el control de seguridad de ventanas deslizantes en la segmentación de los datos; el protocolo Internet Protocol versión 6 (IPv6) [31] en combinación con el protocolo Internet Control Message Protocol versión 6 (ICMPv6) para verificar la conectividad entre los dispositivos en la red y la Orange Pi Lite 2, y, finalmente, la etapa de incorporación de herramientas de expansión de red que son los puertos UBS y tarjetas Ethernet, como se muestra en la Figura 5.

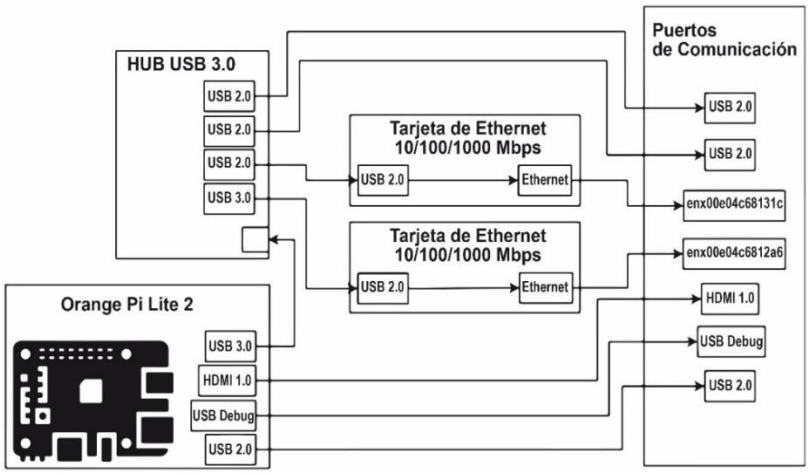

Figura 5. Diseño del sistema de puertos de comunicación. Fuente: elaboración propia.

2.5 Fase 5: funcionamiento del sistema

Se inicia con la integración del sistema de potencia y la incorporación de los puertos USB y Ethernet, después se carga el sistema operativo Armbian por medio de la aplicación balenaEtcher y se instala el boot en una memoria microSD que es insertada en la Orange Pi Lite 2, a la cual se accede por medio de una terminal; desde allí se procede a la configuración de Mininet con los siguientes comandos e instrucciones: apt-get install mininet, apt-get install openvswitch-switch, git clone https://github.com/mininet/mininet/mininet

ymininet/útil/install.sh –fw. Después se crea y emula la red SDN que, a su vez, es soportada con las aplicaciones Open vSwitch [32], GitHub [12] y utilitarios de Mininet, como se observa en la Figura 6.

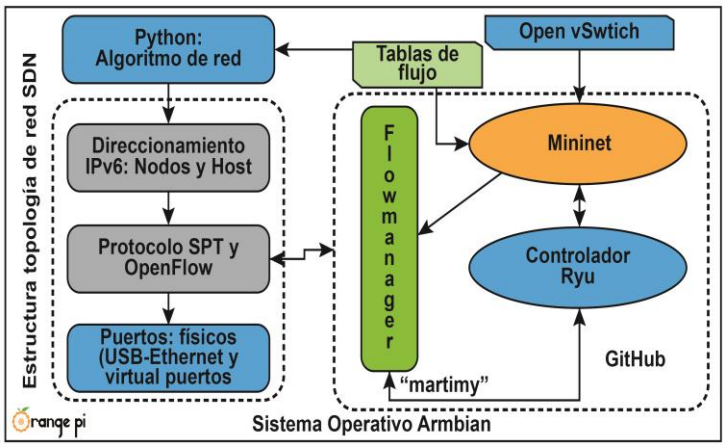

Figura 6. Integración de sistema SDN laboratorio LBRED. Fuente: elaboración propia.

Mediante la ejecución del algoritmo de red en Python (ver Figura 7), se establece la estructura de la topología de red SDN con el uso del protocolo OpenFlow [5] para la gestión de los planos de control; y con la ayuda del Open vSwtich, se establece la comunicación con el controlador Ryu (la instalación del Ryu requiere de las instrucciones: pip install eventlet == 0.30.2, pip install ryu y ryu-manager—observe-link ryu.app.simple_switch). Dicho controlador reenvía los datos y se crean los nodos SDN y hosts de acuerdo con las tablas de flujo; junto con las conexiones Ethernet entre ellos [32].

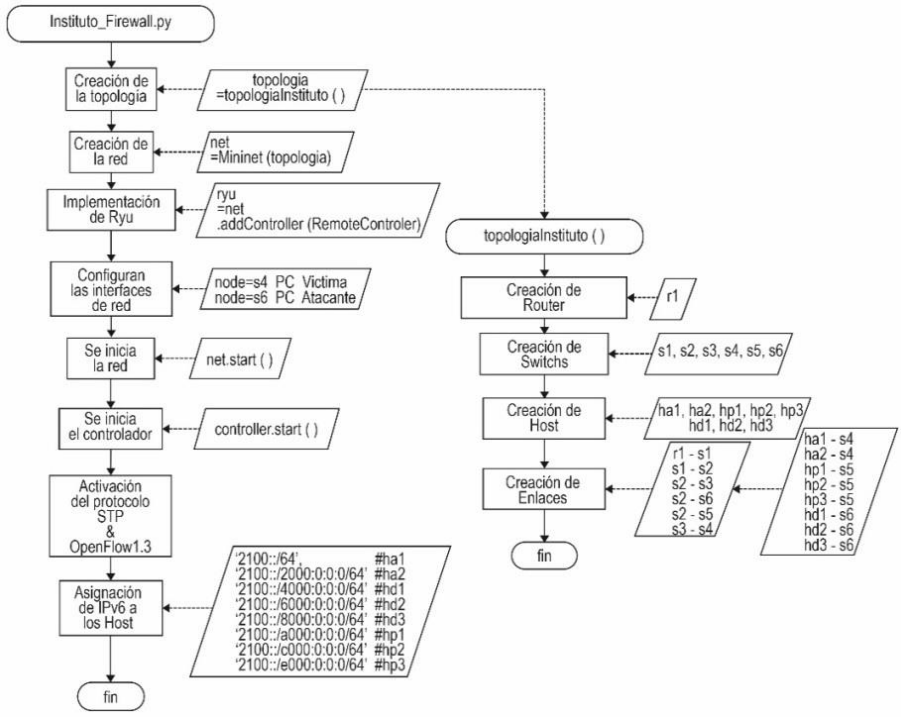

Figura 7. Diagrama de flujo del algoritmo Instituto_Firewall.py. Fuente: elaboración propia.

Una vez establecida la topología, se incorpora a la configuración del Mininet, que gestiona los recursos entre el OpenFlow, el Ryu y los hosts; se procede a establecer de forma simultánea la comunicación por medio de comandos [33]. Después se configuran las interfaces físicas de la red para conectar un equipo externo como host de prueba víctima y otro de atacante para iniciar las pruebas de enrutamiento del tráfico, como se observa en la Figura 7.

A continuación, se describe un script en Python que inicia el proceso mencionado por medio de la librería Mininet.

```
#!/usr/bin/python3
from mininet.net import Mininet
from mininet.node import RemoteController, OVSSwitch
from mininet.link import TCLink
from mininet.cli import CLI
from mininet.log import setLogLevel, info
def sdn_topology():
    setLogLevel('info')
    net = Mininet(controller=RemoteController, link=TCLink, switch=OVSSwitch)
    info('*** Creando el controlador remoto (Ryu)\n')
    ryu_controller = net.addController (
        name='ryuController',
        controller =RemoteController,
        ip='FE80::/10',  # Dirección del host donde corre Ryu
        port=6645        # Puerto OpenFlow por defecto
    )
    Info ('*** Creando hosts\n')
    ha1 = net.addHost('ha1', ip='2100::/64')
    ha2 = net.addHost('ha2', ip='2100::/2000:0:0:0/64')
    hd1 = net.addHost ('hd1', ip='2100::/4000:0:0:0/64')
    hd2 = net.addHost ('hd2', ip='2100::/6000:0:0:0/64')
    info ('*** Creando switch (Open vSwitch)\n')
    s1 = net.addSwitch ('s1')
    info ('*** Creando enlaces\n')
    net.addLink (h1, s1)
    net.addLink(h2, s1)
    info ('*** Iniciando la red\n')
    net.start ()
    info ('*** Verificando conectividad entre hosts\n')
    net.pingAll ()
    info  ('*** Iniciando CLI de Mininet\n')
    CLI (net)
    info ('*** Deteniendo la red\n')
    net.stop ()
if __name__ == '__main__':
    sdn_topology ()
```

El siguiente paso implica la inicialización de la red y el controlador; para ello se activan dos componentes: el protocolo de árbol de expansión (STP) y OpenFlow 1.3. La activación de STP contribuye a la eficiencia y redundancia de la red, asegurando la prevención de bucles y mejorando la estabilidad. Con la implementación de OpenFlow 1.3 se proporciona la comunicación entre el controlador Ryu y los dispositivos de red, para la gestión del flujo de datos con la asignación del direccionamiento IPv6 [34].

Posteriormente se integra el software de Flow Manager mediante los comandos: http://github.com/martimy/flowmanager.git y ryu-manager --obereve--links flowmanager.py [25] Finalmente, en el entorno del laboratorio denominado LBRED con la ayuda del algoritmo martimy [35] se realiza la comunicación con el controlador Ryu para crear y operar la interfaz web de Flow Manager visualizando así el estado de la red.

3. RESULTADOS Y DISCUSIÓN

En el desarrollo de la verificación del funcionamiento de la red SDN con soporte para IPv6 se realizan las comprobaciones básicas por medio del protocolo ICMPv6, con el fin de evidenciar la conectividad para informar y diagnosticar los errores; con esto se comprueba la configuración y el alcance de las tablas de flujo por medio de la tabla de direccionamiento estático IPv6 (ver Tabla 3), de forma que se puedan confinar las pruebas entre bloques de direcciones de los diferentes segmentos de red, como se estableció en la Figura 4, para determinar el comportamiento en tiempo real.

Tabla 3. Tabla de direccionamiento estático IPv6. Fuente: elaboración propia.

Asignación de direccionamiento en la red de pruebas			
#	dl_type	ipv6_src	ipv6_dst
1	IPv6	2100::	2100::1
2	IPv6	2100::1	2100::
3	IPv6	2100::2000:0:0:0	2100::
4	IPv6	2100::2000:0:0:0	2100::1
5	IPv6	2100::1	2100::2000:0:0:0
6	IPv6	2100::a000:0:0:0	2100::c000:0:0:0
7	IPv6	2100::c000:0:0:0	2100::a000:0:0:0
8	IPv6	2100::a000:0:0:0	2100::e000:0:0:0
9	IPv6	2100::e000:0:0:0	2100::a000:0:0:0
10	IPv6	2100::c000:0:0:0	2100::e000:0:0:0
11	IPv6	2100::e000:0:0:0	2100::c000:0:0:0
12	IPv6	2100::4000:0:0:0	2100::6000:0:0:0
13	IPv6	2100::6000:0:0:0	2100::4000:0:0:0
14	IPv6	2100::4000:0:0:0	2100::8000:0:0:0
15	IPv6	2100::8000:0:0:0	2100::4000:0:0:0
16	IPv6	2100::4000:0:0:0	2100::8000:0:0:1
17	IPv6	2100::8000:0:0:1	2100::4000:0:0:0
18	IPv6	2100::6000:0:0:0	2100::8000:0:0:0
19	IPv6	2100::8000:0:0:0	2100::6000:0:0:0
20	IPv6	2100::6000:0:0:0	2100::8000:0:0:1
21	IPv6	2100::8000:0:0:1	2100::6000:0:0:0
22	IPv6	2100::8000:0:0:0	2100::8000:0:0:1
23	IPv6	2100::8000:0:0:1	2100::8000:0:0:0

En la Figura 8 se presentan gráficamente los resultados obtenidos de la forma como se envían los paquetes ICMPv6 entre cada uno de los elementos que hacen parte de la red propuesta con Mininet. De esta forma el Flow Manager se encarga de la detección y gestión del flujo de tráfico de datos a través la virtualización de las comunicaciones en la SDN, y responde de forma interactiva y dinámica a las peticiones dentro de la red. Aquí no solo se hace referencia a las conexiones que son catalogadas como exitosas, también se incluye una perspectiva de comportamiento global de la red en la creación, actualización y restricción de entrada en la tabla de flujo de datos.

En el momento en que se reinicia el controlador Ryu, automáticamente la red SDN activa todos los dispositivos de red; de inmediato el emulador de red Mininet crea la red virtual indicando que el Ryu ya inició, y luego se habilita el Flow Manager para realizar el respaldo de las tablas de flujo.

Posteriormente, se cierra Mininet por un momento y se vuelve a reiniciar, indicándole que no cargue las tablas de flujo, ya que este proceso solo debe realizarse una vez con el respaldo previamente hecho. Tras ejecutar de nuevo Mininet, se restablecen las tablas de flujo utilizando Flow Manager y logrando así una red operativa y funcional.

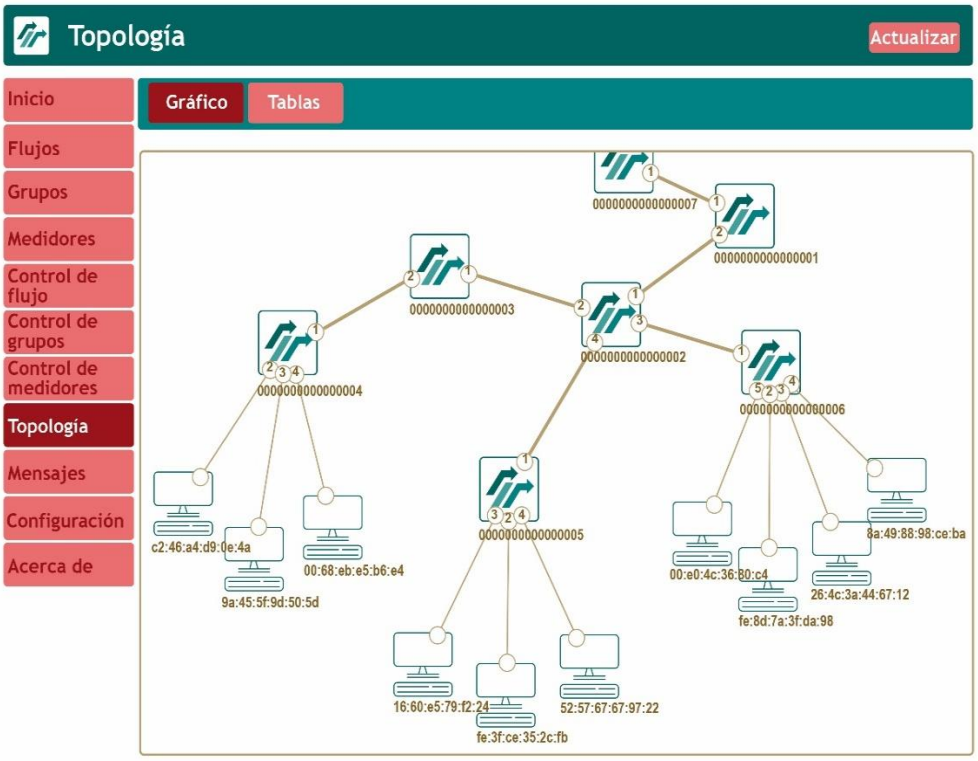

Figura 8. Flow Manager asignando los elementos de red. Fuente: elaboración propia.

Realizando las pruebas sobre la red propuesta en la Figura 4 se puede inferir que: Las pruebas con tráfico externo: se realizan durante 17 minutos en tres eventualidades diferentes, para generar tráfico hacia toda la red con un tamaño de paquetes de 25 bytes de 1000 unidades cada uno y con un ping 1000 veces en cada uno de los hosts y de forma simultánea mediante el fragmento de código de ejemplo, citado a continuación:

```
# Realizar ping a múltiples hosts de forma simultánea
for host in "${hosts[@]}"; do
 Ping -c $numero_pings -s $tamano_paquete -I $interfaz $host &
done
```

Simultáneamente con la red SDN trabajando con tráfico al 100 % se observa que al enviar paquetes entre los equipos externos denominados atacante y víctima al hacer el tránsito desde el exterior pasando por la red SDN, los 25Kbytes de datos llegan completos y las latencias son mínimas, si se comparan cuando hay tráfico y sin tráfico, lo que garantiza una buena velocidad de trasferencia para la red trabajando sobre protocolo IPv6 y el controlador Ryu como se muestra en la Tabla 4.

De igual forma sucede al realizar las pruebas de múltiples paquetes para tres interacciones con las mismas características mencionadas anteriormente, como se muestra en la Figura 9.

Tabla 4. Ejemplo de pruebas de conectividad en la SDN. Fuente: elaboración propia.

Pruebas de ping sin tráfico				
Host	Latencia (ms)	Tiempo máx (ms)	Tiempo mín (ms)	Tasa de transmisión (bytes/s)
Atacante-Victima	0,095	3,059	0,011	26571,639
Víctima - Atacante	2,480	9,074	0,570	484,219
Pruebas de ping con tráfico				
Host	Latencia (ms)	Tiempo máx (ms)	Tiempo mín (ms)	Tasa de transmisión (bytes/s)
Atacante - Victima	0,095	3,059	0,011	26571,639
Víctima - Atacante	2,480	9,074	0,570	484,219

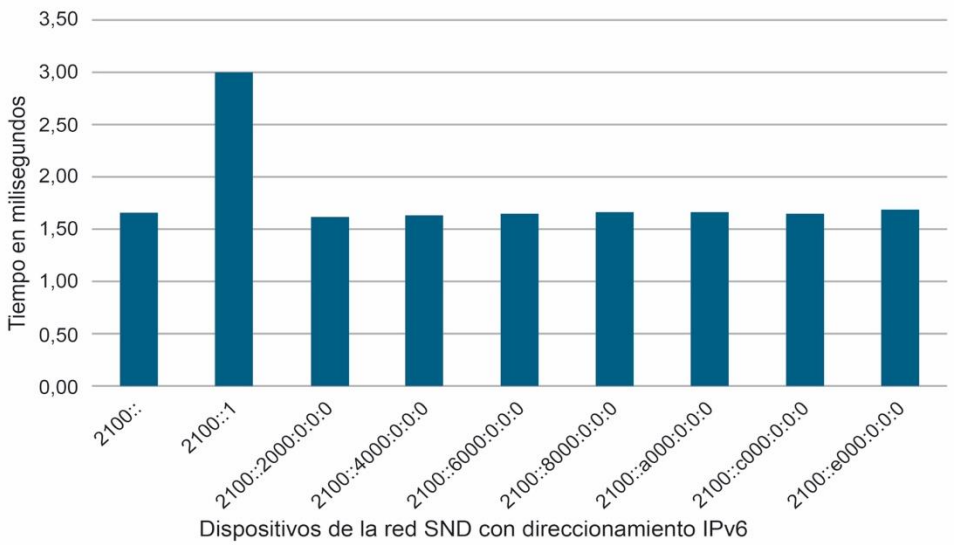

Figura 9. Prueba de envío de paquetes de dispositivo de la red SDN. Tiempo promedio de latencia (ms) para tres escenarios de prueba. Fuente: elaboración propia.

Se observa que al ejecutar simultáneamente el envío de paquetes a cada host el tiempo promedio de duración de los paquetes es de 1,5 milisegundos, excepto el equipo 2100::1 que por control de tráfico presenta un tiempo de 3 milisegundos de latencia.

Al considerar la prueba de envío de paquetes en forma de texto e imagen, tomando como referencia el indicador del número máximo del Hop Limit o límite de salto, con tamaño de 8 bit dentro de la cabecera de IPv6, se establece que cuando el paquete llega al Hop Limit con valor cero, el paquete se descarta, lo cual es una ventaja sobre IPv4, donde dicho trabajo aparentemente lo hace el TTL (tiempo de vida), pero no es tan eficaz como IPv6 para permitir el salto de enrutamiento antes de que se envié el error a través de ICMP y los paquetes se pierdan en el enlace.

En la Tabla 5 se observa el envío del archivo de imagen de 60 Kbyte/s y de texto de 37 Kbyte/s con la duración máxima y mínima a la cual debe llegar el paquete permitiendo una latencia que no causa la pérdida de paquetes y posibilita el envío completo.

Al entrar en activación el Ryu cuando se inicia la red SDN de acuerdo con la Figura 6, el Mininet detecta que el parámetro Hop Limit no alcanza su valor cero para el promedio de la latencia prevista, lo que significa que no hubo pérdida de paquetes en ninguno de los dos sentidos de la comunicación establecida.

Tabla 5. Ejemplo de envío de imágenes y texto. Fuente: elaboración propia.

Archivos	Tiempo promedio de latencia (ms)	Tiempo máx (ms)	Tiempo mín (ms)	Tasa de transmisión (Kbyte/s)
img_m.jpg	0,020	6,400	0,002	60,745
img_s.jpg	0,028	4,829	0,004	61,153
texto_1.txt	0,036	4,521	0,004	38,802
texto_2.txt	0,038	4,950	0,003	36,640

3.1 Detalles del proceso de transmisión

Con el uso del protocolo ICMPv6 como medio para la transmisión de la exfiltración de los paquetes con la técnica de estenografía, se ejecuta la modificación de la dirección MAC para el envío del comando de extracción del archivo de prueba "CC.png" hacia el equipo establecido como víctima. Para tal fin se muestra en la Figura 10 la latencia sufrida en el proceso cuando son enviados los paquetes con un promedio de 0,003 segundos.

Esta técnica afecta directamente la cantidad de información que transporta por paquete, lo cual no significa que sea el determinante de la velocidad de trasmisión. Cuando se hace el cambio por direccionamiento MAC con 40 bit por stegopaquete, se logran velocidades de 12,160 Kbps. De igual forma, con la técnica estenográfica por cambio del Hop Limit con el uso de 8bits por stegopaquete, la velocidad alcanzada es de 2,430 Kbps. Estas cifras proporcionan perspectivas valiosas sobre las capacidades y limitaciones de las técnicas implementadas en el contexto de seguridad en redes SDN.

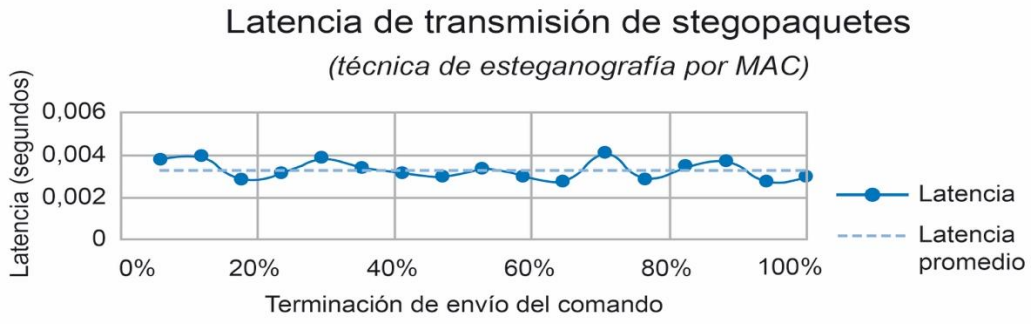

Figura 10. Variación de la latencia de transmisión del stegopaquetes. Se observa la latencia que tiene en llegar del transmisor al receptor los stegopaquetes. Fuente: elaboración propia.

Si tomamos un referente reciente en lo que respecta a redes SDN y sus diversas adopciones [36], se verifica la facilidad de implementación de este tipo de redes y se abordan los fundamentos partiendo de lo que es el desacoplamiento del plano de control y el plano de datos y haciendo un análisis de los controladores ODL, Ryu y ONOS; se observa que en el caso de la investigación propuesta, al implementar el controlador Ryu hay una mejora en los parámetros de: rendimiento, seguridad, pruebas de velocidad y ancho de banda, que también analizó [36] en su investigación.

3.2 Análisis del ancho de banda

El análisis del ancho de banda en el laboratorio SDN, específicamente con relación a las técnicas de esteganografía implementadas, reveló notables diferencias entre las metodologías utilizadas. Al realizar el enfoque en la técnica de esteganografía por cambio de dirección MAC, la Figura 11 presenta la variación en la latencia entre stegopaquetes, que representa el tiempo transcurrido entre el envío de un stegopaquete y el siguiente.

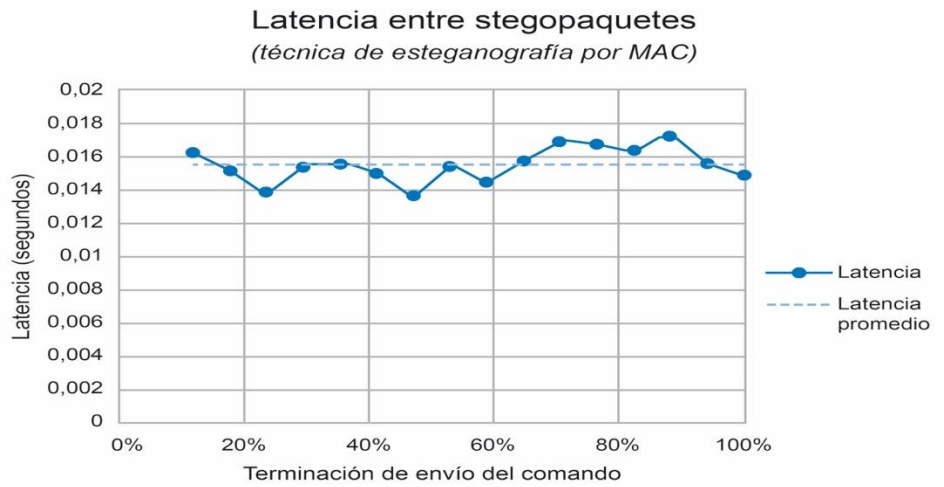

Figura 11. Variación de la latencia entre stegopaquetes en el envío de comando. Se analiza la latencia que tiene el transmisor en enviar un stegopaquetes. Fuente: elaboración propia.

El promedio alcanzado en la traza de pruebas fue de 0,025 segundos. Considerando que cada stegopaquete contiene 40 bits, el análisis de ancho de banda utilizando el algoritmo de control asociado a esta técnica arrojó una velocidad de 2.577 Kbps.

Al analizar la Figura 12, asociada a la técnica de modificación por Hop Limit, se lograron identificar picos de latencias altos, con valores de 3,460 segundos durante el 70 % del envío de los archivos “CC.png”. Sin embargo, la latencia promedio se mantiene baja, por debajo de los 0,048 segundos aproximadamente, aunque en términos generales no afecta la transmisión de los paquetes y la comunicación es estable. En las pruebas se detalla que el ancho de banda registrado es de 171 bps, un valor que es bajo y funcional para los casos en que se exfiltran los datos.

Los hallazgos evidencian que es importante fijar las políticas de seguridad de forma más restrictiva en este tipo de ambientes. Para ello se debe proceder principalmente con la creación de filtros en el firewall, no solo en el direccionamiento con IPv6, sino también sobre el direccionamiento MAC. De esta forma se logra mitigar ataques basados en técnicas de esteganografía, como se realizó en este laboratorio de pruebas.

Así mismo, se recomienda deshabilitar el protocolo ICMPv6 de no ser necesario; como es el caso cuando se crean un número considerables de host o computadoras dentro de la red, ya que aumenta el riesgo de seguridad siendo un vector potencial de ataque si se mantiene habilitado innecesariamente.

La técnica de estenografía por Hop Limit la hace muy fácil y versátil para ambientes de compatibilidad con el protocolo IPv6, por ello es conveniente la capacitación del personal técnico de seguridad para prevenir la inyección de stegomalware, especialmente en la primera línea de seguridad en el núcleo de la red.

El valor agregado de la investigación se centra en una solución práctica, económica y tangible en la implementación de redes SDN física, como un laboratorio de pruebas con interfaces para la evaluación del comportamiento de conectividad bajo el protocolo IPv6, que la hace diferente a una red simulada. Por otra parte, las limitaciones encontradas en el desarrollo de la investigación están en los controladores y su compatibilidad con el hardware y el software existentes en el mercado, pues se requiere de configuraciones avanzadas y estudios de documentos técnicos y de programación, también de la adquisición de dispositivos con capacidades robustas y avanzadas de procesamiento.

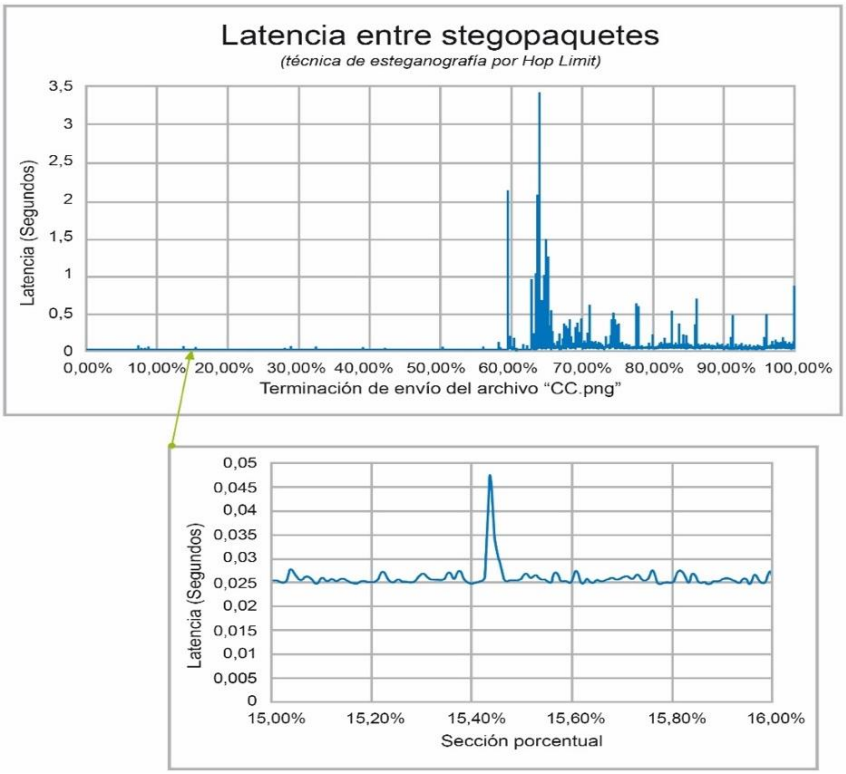

Figura 12. Comportamiento del retraso de los stegopaquetes del archivo a exfiltrar en la transmisión de paquetes. Fuente: elaboración propia.

En lo que respecta a las implicaciones prácticas, es una herramienta útil para ingenieros de redes, investigadores y desarrolladores que busquen validar arquitecturas SDN con soporte nativo de IPv6. En la implementación se facilitan las pruebas para analizar posibles fallos de conectividad, evaluar rendimiento y verificar la eficiencia del enrutamiento y segmentación con IPv6, por medio de la combinación de controladores específicos para este tipo de redes en entornos de experimentación académica.

La aplicación de la metodología Top-Down en el diseño y la implementación del laboratorio SDN/IPv6 denominado LBRED permitió alcanzar los objetivos con un enfoque estructurado para identificar los requisitos técnicos, seleccionar los componentes y consolidar un entorno de pruebas en un entorno real, comparado con [4], [5] y [10] que solo orientan las investigaciones en la revisión sistemática y emulación.

El dispositivo se implementó físicamente y es significativo respecto a los estudios previos, como es el caso de [13] y [15], que desarrollaron el modelado y la simulación de SDN/IPv6. El LBRED permite realizar pruebas reales, lo que mejora la precisión de las mediciones y su replicabilidad en entornos educativos y de validación experimental. Este aporte práctico refuerza el vínculo entre la investigación académica y las aplicaciones reales en ingeniería de redes.

Los resultados experimentales evidenciaron que el sistema mantiene un consumo estable de recursos por debajo del 11 % de CPU y 23 % de RAM, incluso durante las pruebas de seguridad con exfiltración de datos mediante técnicas de esteganografía (por cambio de MAC y Hop Limit). Dichos hallazgos confirman la viabilidad del diseño para analizar comportamientos de tráfico y mecanismos de ciberseguridad en redes IPv6 bajo control SDN. En contraste con la seguridad en SDN, [19] y [21] se enfocaron en el uso de IA para la automatización en ciberseguridad y [20] y [3] realizan la revisión exhaustiva de IA y automatización.

4. CONCLUSIONES

La incorporación de IPv6 demostró una gestión de red más dinámica, con asignación flexible del ancho de banda y priorización de tráfico en tiempo real. La arquitectura SDN implementada favorece la escalabilidad, reducción de costos operativos y actualización del sistema sin modificar el hardware para su adopción en laboratorios académicos y entornos empresariales pequeños.

Se comprobó que la utilización del protocolo IPv6 es compatible y permite la transición para este tipo de redes convergentes, porque la gestión centralizada facilita el control y monitoreo en tiempo real, como se comprobó con el indicador mínimo de salto en la asignación del ancho de banda eficiente y la prioridad del tráfico.

Una característica destacable del sistema fue su flexibilidad y escalabilidad, evidenciada en la capacidad de actualizar y modificar la red mediante el uso del Flow Manager, sin requerir cambios en el hardware. Esta funcionalidad posibilita la gestión centralizada de múltiples dispositivos desde un único controlador.

Una de las capacidades más relevantes observadas en la red SDN radicó en su interoperabilidad y adopción de estándares abiertos. Con esta característica se garantizó la compatibilidad entre distintos tipos de hardware durante la integración de dispositivos y plataformas, lo que evidencia la madurez tecnológica y la versatilidad del modelo implementado.

Se demostró que con las tablas de flujo se facilita el proceso de envío de paquetes soportado con la interfaz de aplicación del Flow Manager; por medio de este se detecta el tráfico y la forma como la red crece dinámicamente a medida que se agregan más dispositivos y le comunica al controlar Ryu para reducir la latencia y mejorar el desempeño de la red SDN.

Una deducción valiosa en la implementación de redes SDN es la facilidad de integrar el protocolo IPv6 con miras unir redes de infraestructuras híbridas para desarrollar la combinación de diversas aplicaciones y servicios mediante pruebas de validación.

Por último, este trabajo contribuye a fortalecer el conocimiento y la visibilidad de SDN en el contexto colombiano. De acuerdo con [8], aún existe una limitada difusión de esta tecnología y se requiere profundizar su estudio y aplicabilidad; por tanto, la herramienta diseñada sirve para la enseñanza, la investigación aplicada y la validación experimental, promoviendo el desarrollo de capacidades locales en ingeniería de redes y ciberseguridad.

5. AGRADECIMIENTO Y FINANCIACIÓN

Este trabajo se realizó como producto de la investigación entre los autores como miembros de Grupo de Investigación en Telecomunicaciones y Nuevas Tecnología (GITENT) de la Universidad de Pamplona del Programa Ingeniería en Telecomunicaciones, con recursos propios, sin financiamiento institucional. Se le agradece a la institución por facilitar los espacios donde se desarrollaron las pruebas.

6. REFERENCIAS

[1] M. Hussain, N. Shah, R. Amin, S. S. Alshamrani, A. Alotaibi, and S. Mohsan Raza, “Software-Defined Networking: Categories, Analysis, and Future Directions,” *Sensors*, vol. 22, no. 15, p. 5551, Jul. 2022. https://doi.org/10.3390/s22155551

[2] H. A. Al-Gboury, and S. A. Al-Talib, “Investigate and Compare Software-Defined Network Controllers for UAV Networks Management,” *IOP Conf. Ser. Mater. Sci. Eng.*, Jul. 15-16, 2020, vol. 928, p. 022055. https://doi.org/10.1088/1757-899X/928/2/022055

[3] G. Lakhani, and A. Kothari, “Fault Administration by Load Balancing in Distributed SDN Controller: A Review,” *Wireless Pers. Commun.*, vol. 114, no. 4, pp. 3507-3539, Oct. 2020.

https://doi.org/10.1007/s11277-020-07545-2
[4] K. Marszałek, and A. Domański, "A fluid flow model for the software defined wide area networks analysis," *Sci. Rep.*, vol. 15, no. 3713, pp. 1-18, Jan. 2025. https://doi.org./10.1038/s41598-025-88162-6
[5] B. W. Oviedo Bayas, E. R. Zhuma Vera, G. K. Bowen Calero, and B. S. Patiño Maisanche, "Implementación de una red definida por software que permita brindar servicio de Voip seguros," *Rev. Univ. Socied.*, vol. 13, no. 2, pp. 389-396, Mar.-Apr. 2021. http://scielo.sld.cu/scielo.php?script=sci_abstract&pid=S2218-36202021000200389&lng=es&nrm=iso
[6] D. Haro Mendoza, L. Tello Oquendo, and L. A. Marrone, "A comparative evaluation of the performance of open-source SDN controllers," *Lajc*, vol. 7, no. 2, pp. 64-77, Dec. 2020. https://lajc.epn.edu.ec/index.php/LAJC/article/view/218
[7] F. J. Badaró V. Neto, C. J. Miguel, A. C. dos S. de Jesus, and P. N. M. Sampaio, "SDN Controllers - A Comparative approach to Market Trends," in *9th Int. Workshop ADVANCEs in ICT Infrastr. Serv. (ADVANCE 2021)*, Zaragoza, Spain, Feb. 2021, vol. 16, pp. 48-51. https://doi.org/10.48545/advance2021-shortpapers-3
[8] J. E. Cáceres Guevara, and C. A. Casilimas Fajardo, "Arquitectura y funcionamiento de redes definidas por software (SDN)," Tesis de especialización, Universidad Distrital Francisco Jose de Caldas, Bogotá, Colombia, 2021. http://hdl.handle.net/11349/29727
[9] B. Birhauni Deneke, A. Mulatu Beyene, and E. Ayenew Haile, "Improving Software Defined Network controllers in a multi-vendor environment," *Heliyon*, vol. 10, no. 4, p. e26215, Feb. 2024. https://doi.org/10.1016/j.heliyon.2024.e26215
[10] B. Sapkota, B. R. Dawadi, and S. R. Joshi, "Controller placement problem during SDN deployment in the ISP/Telco networks: A survey," *Eng. Reports*, vol. 6, no. 2, p. e12801, Feb. 2024. https://doi.org/10.1002/eng2.12801
[11] C. Urrea, and D. Ben, "Software-Defined Networking Solutions, Architecture and Controllers for the Industrial Internet of Things : A Review," *Sensors*, vol. 21, no. 19, p. 6585, Oct. 2021. https://doi.org/10.3390/s21196585
[12] A. A. Zopellaro Soares *et al.*, "SDN-based teleprotection and control power systems: A study of available controllers and their suitability," *Int. J. Netw. Manag.*, vol. 31, no. 3, p. e2112, May.-Jun. 2021. https://doi.org/10.1002/nem.2112
[13] D. H. Sim, J. Shin, and M. H. Kim, "Software-Defined Networking Orchestration for Interoperable Key Management of Quantum Key Distribution Networks," *Entropy*, vol. 25, no. 6, p. 943, Jun. 2023. https://doi.org/10.3390/e25060943
[14] E. Zhuma Mera, D. M. Guzmán Vélez, C. A. Cáceres Miranda, and B. W. Oviedo Bayas, *Análisis De Las Redes Definidas Por Software (Sdn) Frente a Redes Tcp/Ip Y Combinadas*. Guayaquil, Ecu: Compas, 2020.
[15] M. N. Amin Sheikh, I. S. Hwang, M. Saibtan Raza, and M. Syuhaimi Ab-Rahman, "A Qualitative and Comparative Performance Assessment of Logically Centralized SDN Controllers via Mininet Emulator," *Computers*, vol. 13, no. 4, p. 85, Mar. 2024. https://doi.org/10.3390/computers13040085
[16] S. Midha *et al.*, "A Secure Multi-factor Authentication Protocol for Healthcare Services Using Cloud-based SDN," *Comput. Mater. Contin.*, vol. 74, no. 2, pp. 3711-3726, Oct. 2023. https://doi.org/10.32604/cmc.2023.027992
[17] M. A. Al-Shareeda, S. Manickam, M. A. Saare, and N. Bin Omar, "Sadetection: Security Mechanisms to Detect SLAAC Attack in IPv6 Link-Local Network," *Inform.*, vol. 46, no. 9, pp. 31-38, Oct. 2022. https://doi.org/10.31449/inf.v46i9.4441
[18] S. Faezi, and A. A. Shirmarz, "A Comprehensive Survey on Machine Learning using in Software Defined Networks (SDN)," *Hum-Cent. Intell. Syst.*, vol. 3, no. 3, pp. 312-343, Sept. 2023. https://doi.org/10.1007/s44230-023-00025-3
[19] Y. Bousnah, Y. Baddi, F. Bensalah, and F. Hassani, "Artificial Intelligence in Software-Defined Networks Security: A Survey," *Procedia Comput. Sci.*, vol. 265, pp. 554-559. 2025. https://doi.org/10.1016/j.procs.2025.07.218
[20] D. D. Fuentes-Doria, A. E. Toscano-Hernández, E. Malvaceda-Espinoza, J. L. Díaz Ballesteros, and L. Díaz Pertuz, *Metodología de la investigación: conceptos, herramientas y ejercicios prácticos en las ciencias administrativas y contables*. Medellín, Col: Editorial Universidad Pontificia Bolivariana, 2020.
[21] J. L. Arias Gonzáles, and M. Covinos Gallardo, *Diseño y metodología de la investigación*. Arequipa, Per: Enfonques Consulting EIRL, 2021.
[22] A. Shirmarz, and A. Ghaffari, "Performance issues and solutions in SDN-based data center: a survey," *J. Supercomput.*, vol. 76, no. 10, 7545–7593, Oct. 2020. https://doi.org/10.1007/s11227-020-03180-7

[23] P. J. Cairo Martínez, "Design Procedure for Business Wi-Fi Networks," *Telemática*, vol. 22, pp. 98-109, Sep. 2024. https://revistatelematica.cujae.edu.cu/index.php/tele/article/view/620
[24] X. Zhang, L. Cui, K. Wei, F. Po Tso, Y. Ji, and W. Jia, "A survey on stateful data plane in software defined networks," *Comput. Networks*, vol. 184, p. 107597, Jan. 2021. https://doi.org/10.1016/j.comnet.2020.107597
[25] Z. Latif, K. Sharif, F. Li, M. M. Karim, S. Biswas, and Y. Wang, "A comprehensive survey of interface protocols for software defined networks," *J. Netw. Comput. Appl.*, vol. 156, p. 102563, Apr. 2020. https://doi.org/10.1016/j.jnca.2020.10256ma3
[26] V. A. Shirsath, and M. M. Chandane, "Beyond the Basics: An In-Depth Analysis and Multidimensional Survey of Programmable Switch in Software-Defined Networking," *Int. J. Networked Distrib. Comput.*, vol. 13, no. 8, pp. 1-27, Dec. 2024. https://doi.org/10.1007/s44227-024-00049-6
[27] M. Koster. *Vecta.io Powerful, online SVG editor for teams.* Siemens Industry Software. 2025. Holland. Accessed: Apr. 11, 2025. [Online]. Available: https://vecta.io/
[28] I. Koulouras, I. Bobotsaris, S. V. Margariti, E. Stergiou, and C. Stylios, "Assessment of SDN Controllers in Wireless Environment Using a Multi-Criteria Technique," *Inf.*, vol. 14, no. 9, p. 476, Aug. 2023. https://doi.org/10.3390/info14090476
[29] A. D. Ferguson *et al.*, "Orion: Google's Software-Defined Networking Control Plane," presented at *Proceed. 18th USENIX Symp. Netw. Syst. Design Implement.*, Apr. 12-14, 2021. [Online]. Available: https://www.usenix.org/conference/nsdi21/presentation/ferguson
[30] S. Askar, and F. Keti, "Performance Evaluation of Different SDN Controllers: A Review," *Int. J. Sci. Bus.*, vol. 5, no. 6, pp. 67-80, May. 2021. https://doi.org/10.5281/zenodo.4742771
[31] A. Abubakar Ibrahim, R. A. Abdulmolla Abdulghafor, and S. Wani, "A New Concept of Duplicate Address Detection Processes in IPv6 Link-Local Network," *Int. J. Innov. Comput.*, vol. 12, no. 2, pp. 9-16, Nov. 2022. https://doi.org/10.11113/ijic.v12n2.368
[32] R. Wazirali, R. Ahmad, and S. Alhiyari, "Sdn-openflow topology discovery: An overview of performance issues," *Appl. Sci.*, vol. 11, no. 15, p. 6999, Jul. 2021. https://doi.org/10.3390/app11156999
[33] T. Adhikari, A. Kumar Khan, and M. Kule, "An analytical review of security issues in centralized and distributed SDN environments," *Inf. Secur. J.*, vol. 34, no. 6, pp. 713-743, Aug. 2025. https://doi.org/10.1080/19393555.2025.2550762
[34] E. Alotaibi, "A tutorial on software-defined networks emulation," *J. Eng. Res.*, vol. 13, no. 2, pp. 666-673, Jun. 2025. https://doi.org/10.1016/j.jer.2023.12.005
[35] E. L. Fernandes *et al.*, "The road to BOFUSS: The basic OpenFlow userspace software switch," *J. Netw. Comput. Appl.*, vol. 165, p. 102685, Sep. 2020. https://doi.org/10.1016/j.jnca.2020.102685
[36] G. Salazar-Chacón, "Hybrid Networking SDN and SD-WAN : Traditional Network Architectures and Software-Defined Networks Interoperability in digitization era," *J. Comp. Sci. Technol.*, vol. 22, no. 1, p. e07, Apr. 2022. https://doi.org/10.24215/16666038.22.e07

CONFLICTO DE INTERÉS

Los autores manifiestan que no existe ningún conflicto de interés, ya sea financiero, profesional o personal, que pudiera surgir de la publicación del presente artículo.

CONTRIBUCIÓN DE AUTORÍA

Jorge E. Herrera Rubio: concepción, redacción, análisis de información, metodología, redacción y elaboración del artículo.

Luz María Ballesteros: concepción, redacción, análisis de información, laboratorio, desarrollo del software, configuración del hardware, pruebas, gestión de la información, elaboración y creación de imágenes, adaptación de tecnologías de bajo costo para el aprendizaje de SDN, implementación de IPv6 en redes de nueva generación, comproboración de la gestión centralizada facilita el control y monitoreo en tiempo real, la asignación del ancho de banda y la prioridad del tráfico.